Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

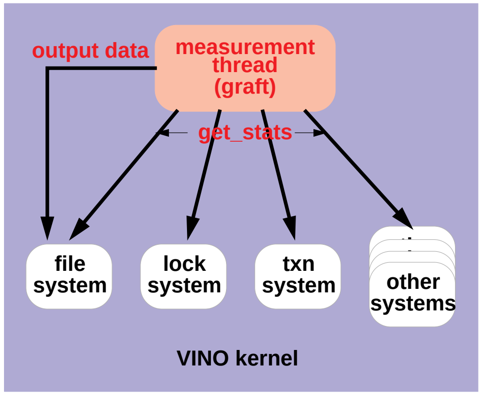
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

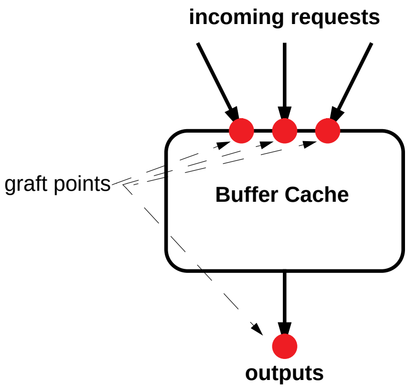
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

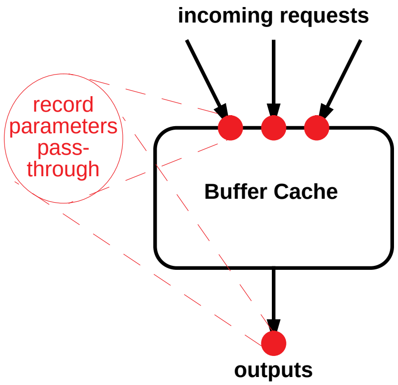
Measurement Thread



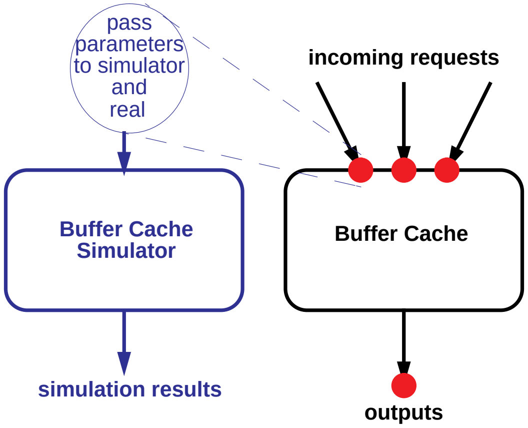
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

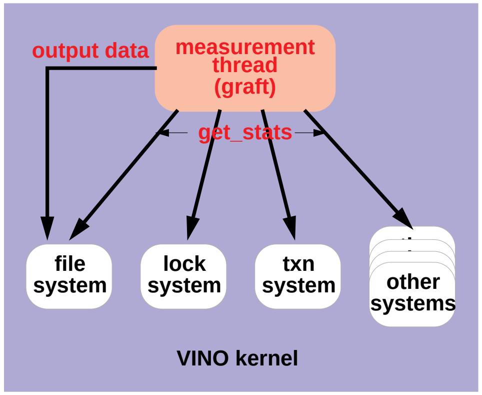
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

Measurement Thread



output data

measurement thread (graft)

get_stats

file system

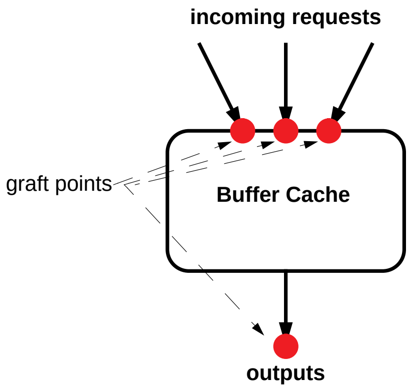
lock system

txn system

other systems

VINO kernel

Generating Traces and Logs



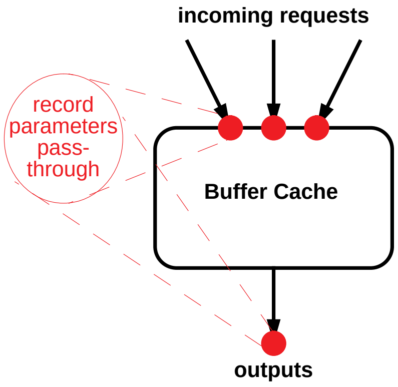
incoming requests

graft points

Buffer Cache

outputs

Generating Traces and Logs



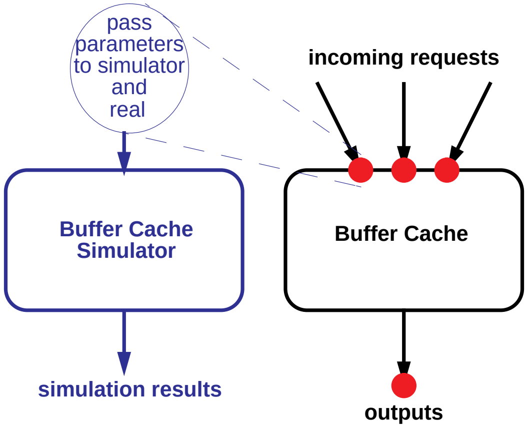
incoming requests

record parameters pass-through

Buffer Cache

outputs

In-Situ Simulation



pass parameters to simulator and real

incoming requests

Buffer Cache Simulator

Buffer Cache

simulation results

outputs

What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

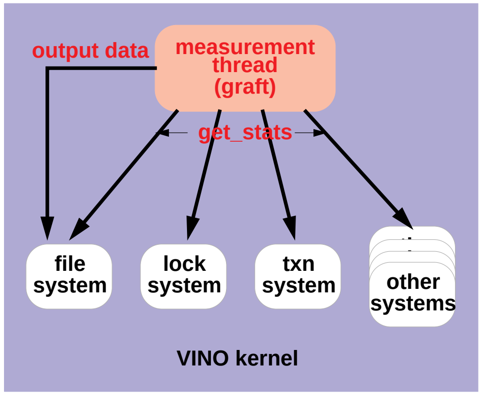
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

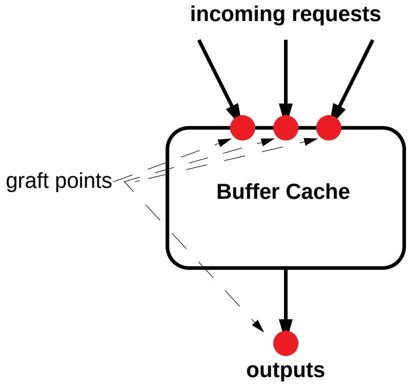
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

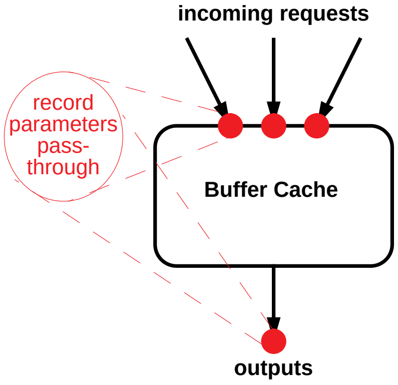
Measurement Thread



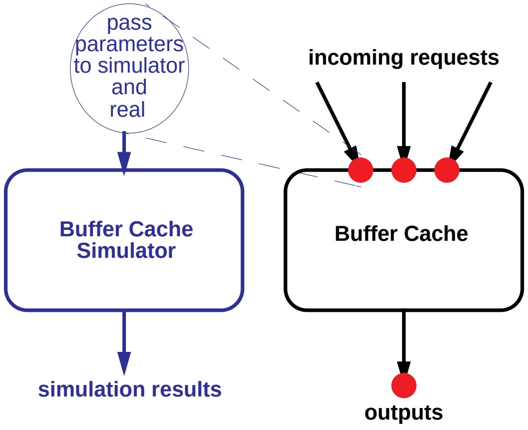
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

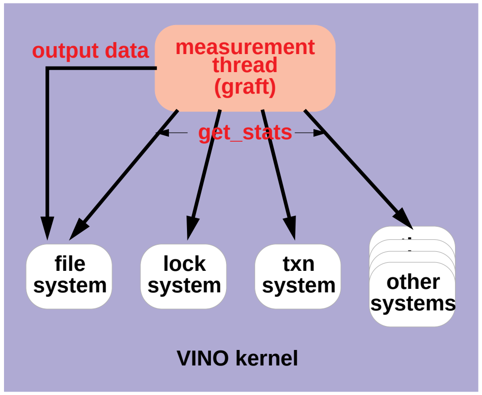
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

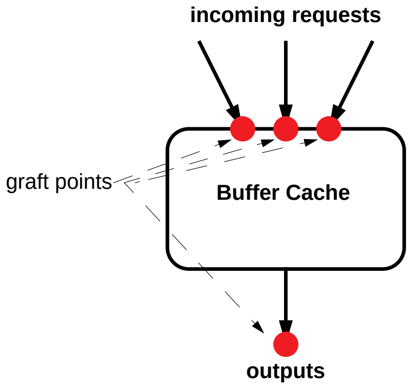
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

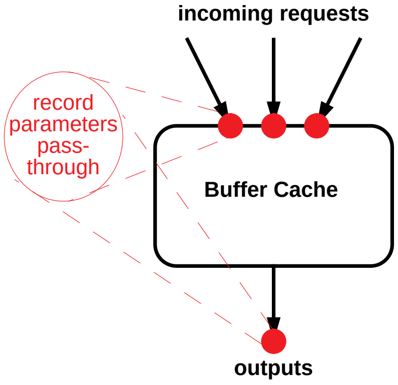
Measurement Thread



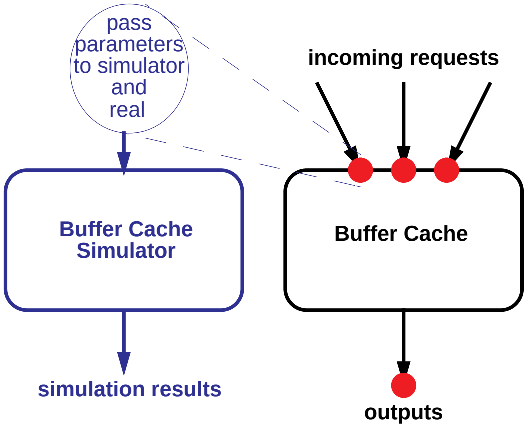
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

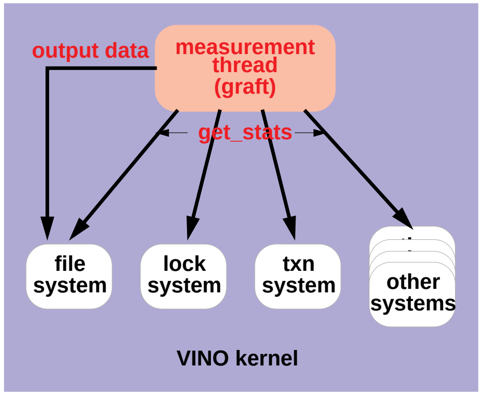
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

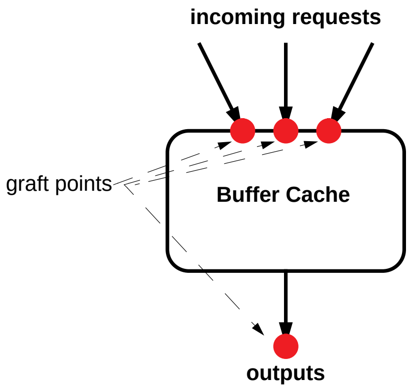
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

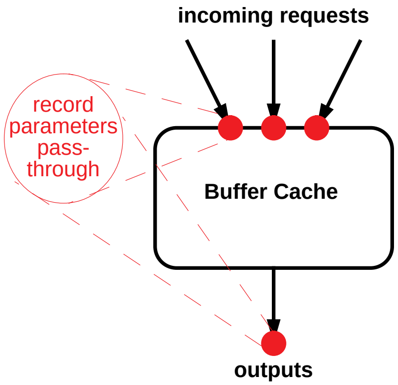
Measurement Thread



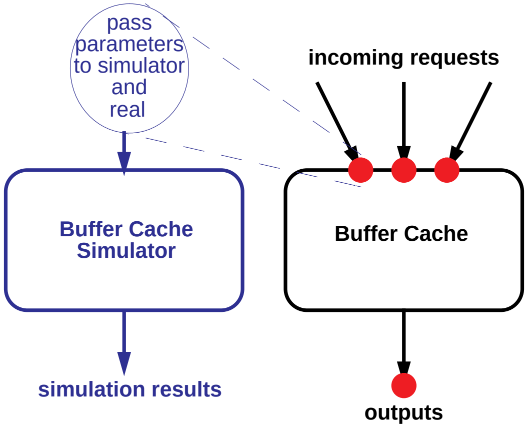
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

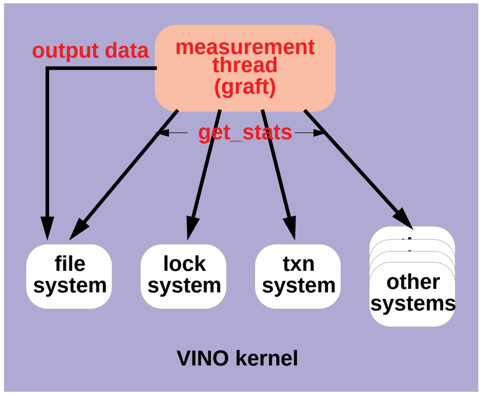
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

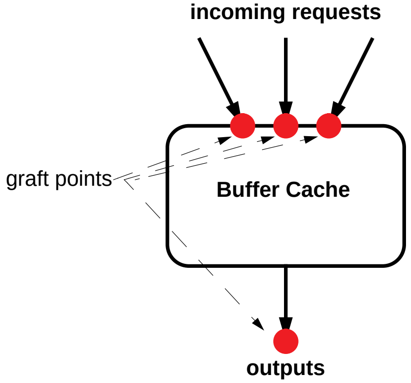
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

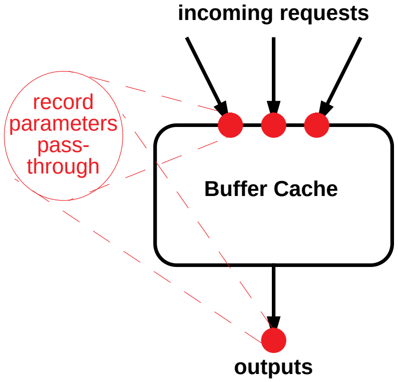
Measurement Thread


output data
measurement thread (graft)
get_stats
file system
lock system
txn system
other systems
VINO kernel

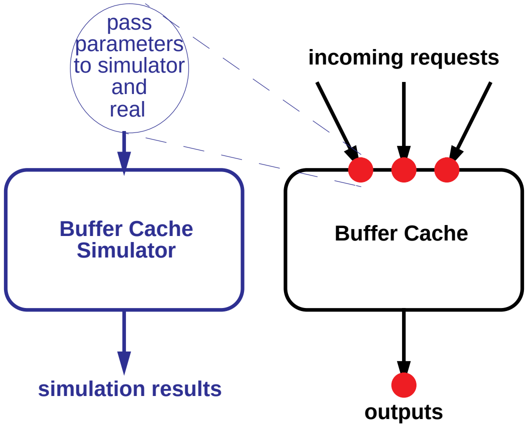
Generating Traces and Logs


incoming requests
graft points
Buffer Cache
outputs

Generating Traces and Logs


incoming requests
record parameters pass-through
Buffer Cache
outputs

In-Situ Simulation


pass parameters to simulator and real
incoming requests
Buffer Cache Simulator
Buffer Cache
simulation results
outputs

What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

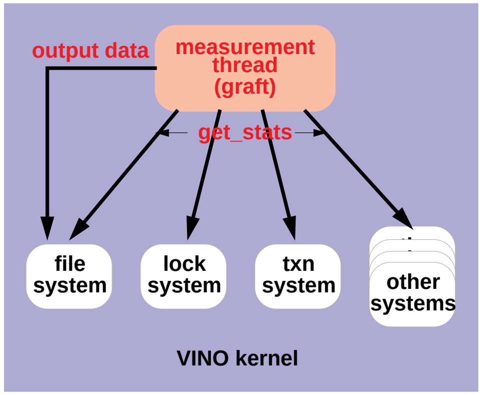
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

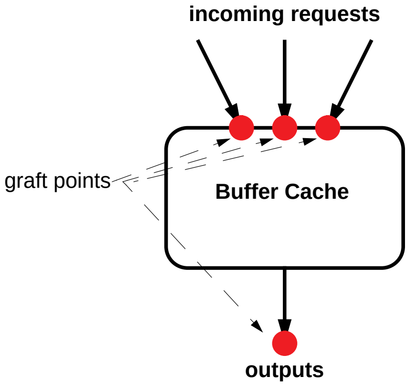
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

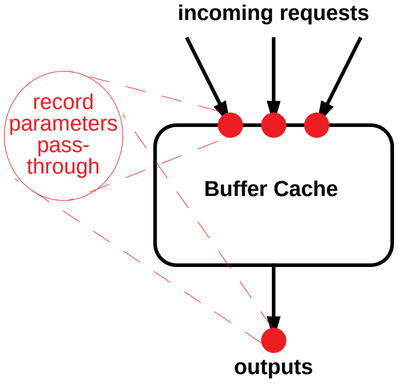
Measurement Thread



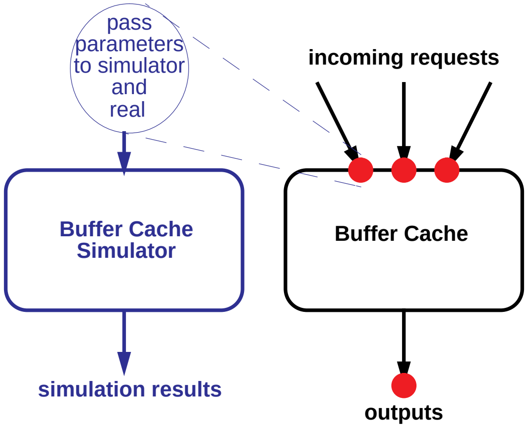
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

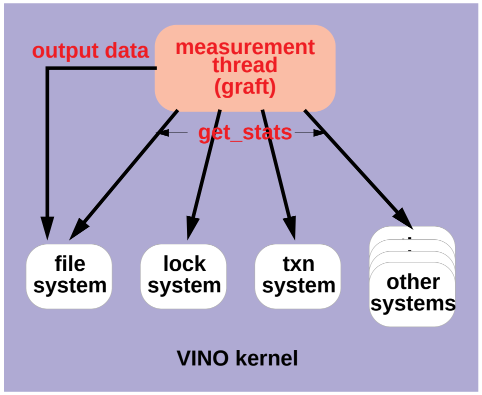
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

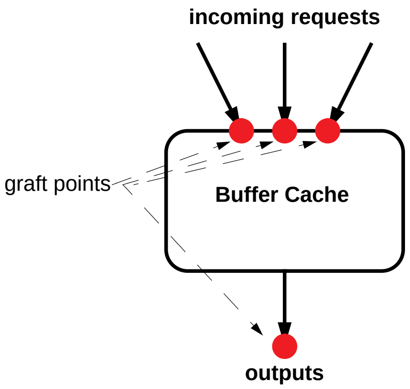
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

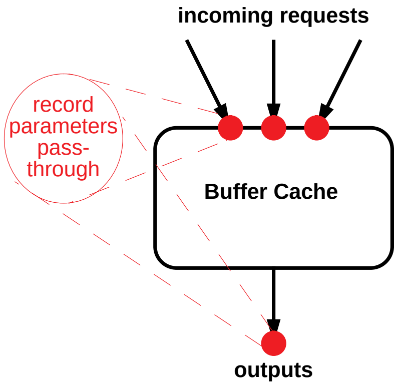
Measurement Thread



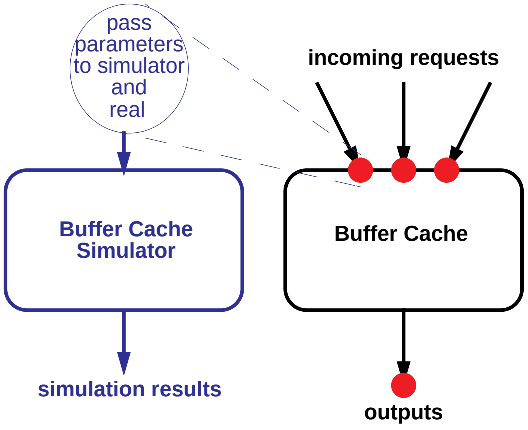
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

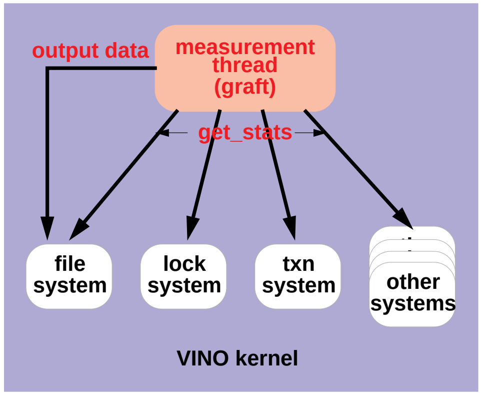
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

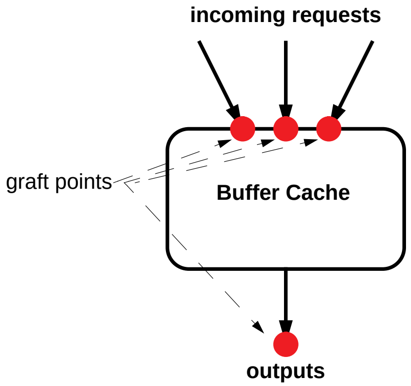
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

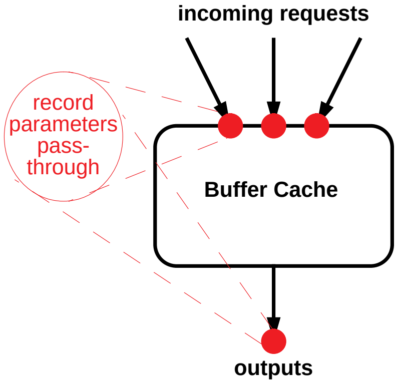
Measurement Thread



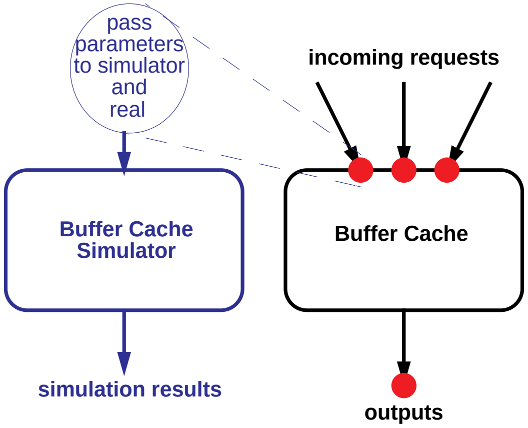
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

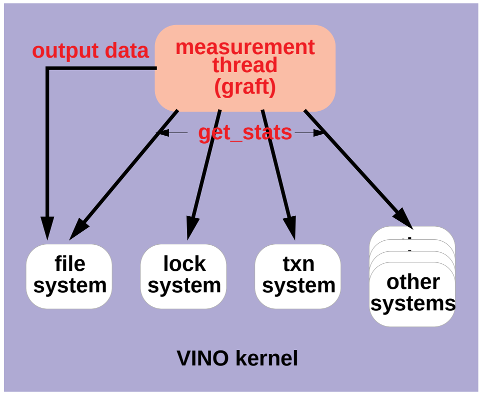
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

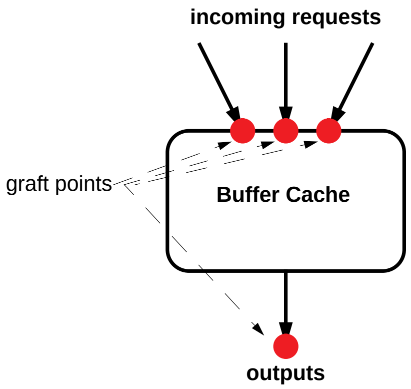
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

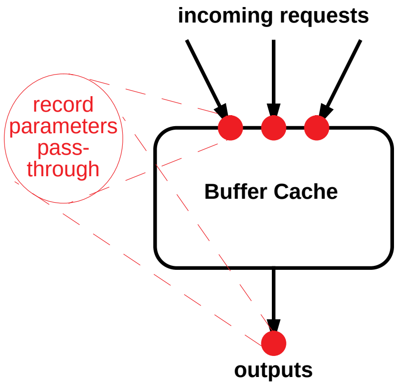
Measurement Thread



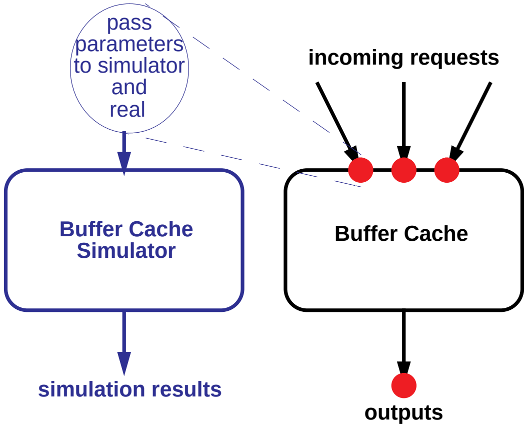
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

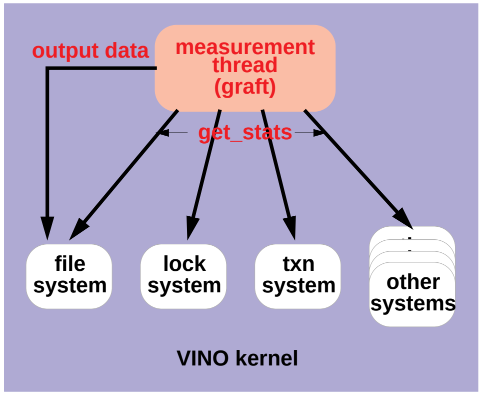
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

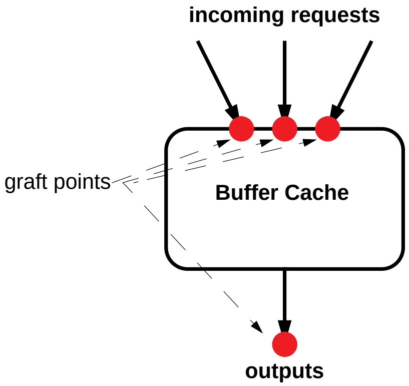
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

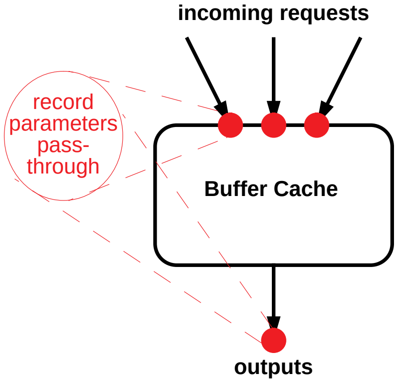
Measurement Thread



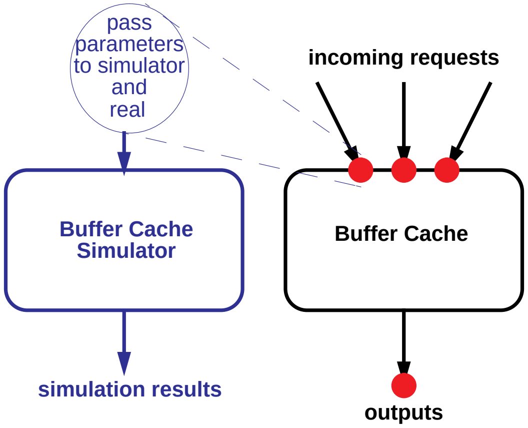
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

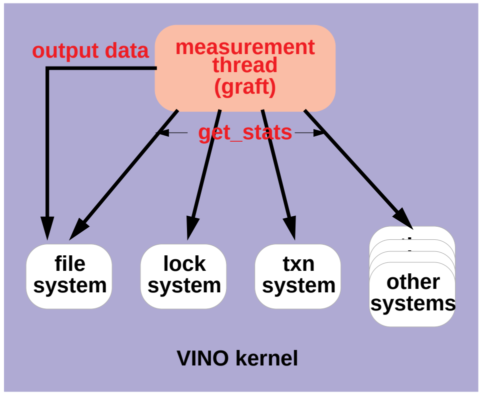
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

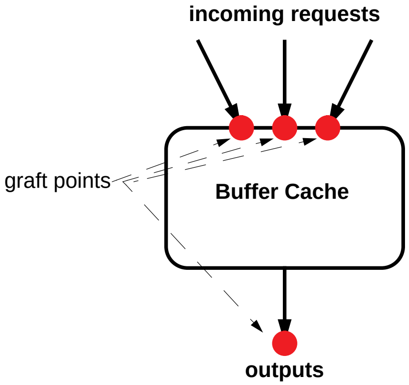
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

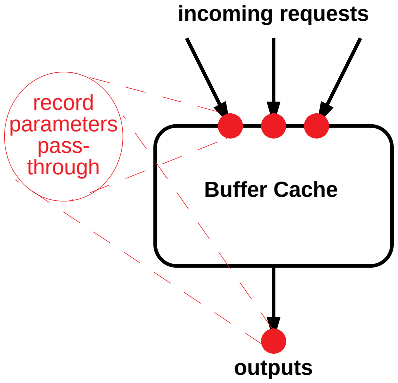
Measurement Thread



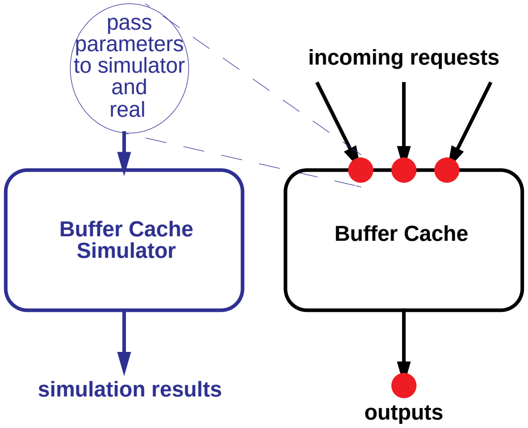
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

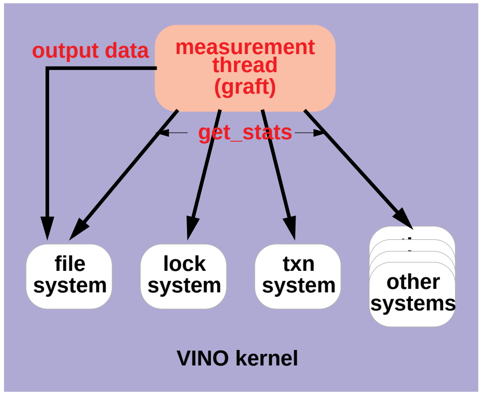
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

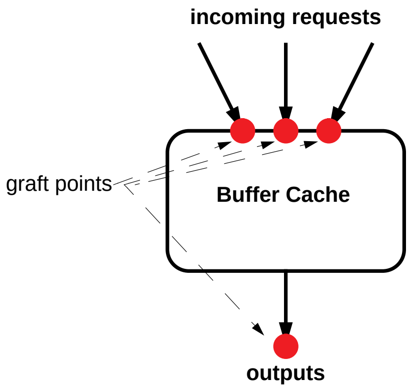
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

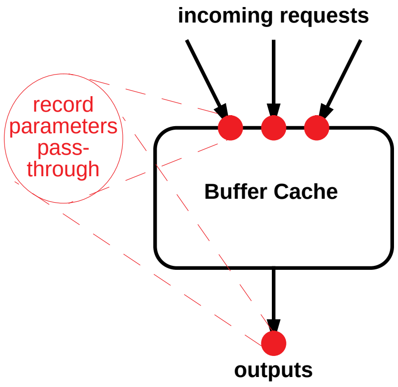
Measurement Thread



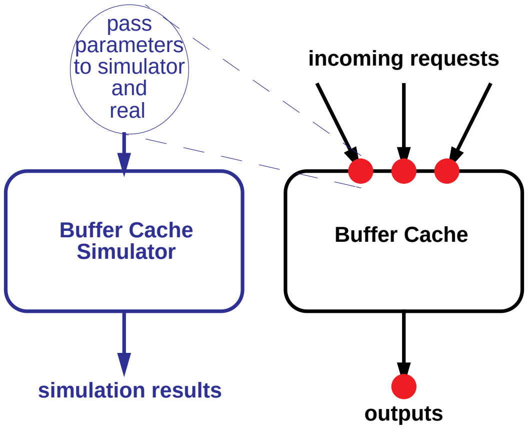
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

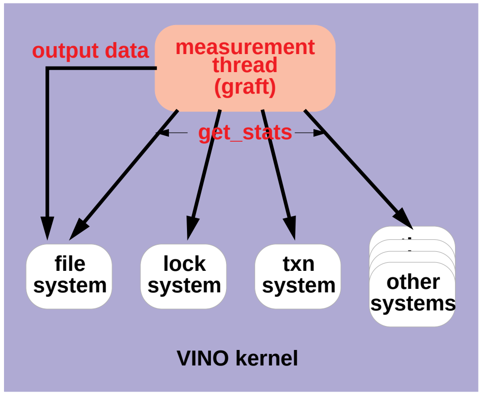
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

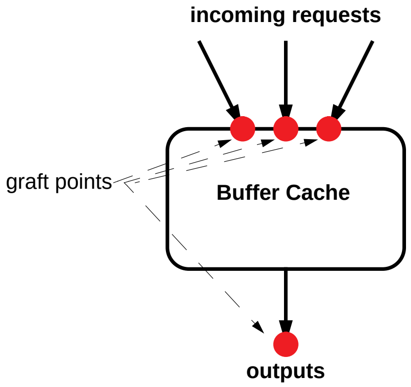
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

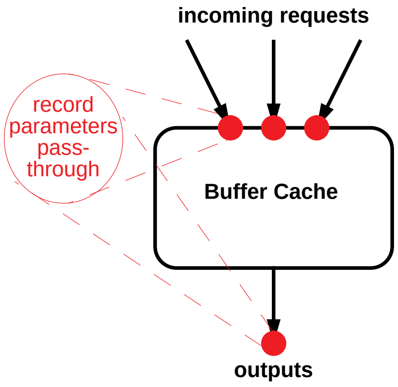
Measurement Thread



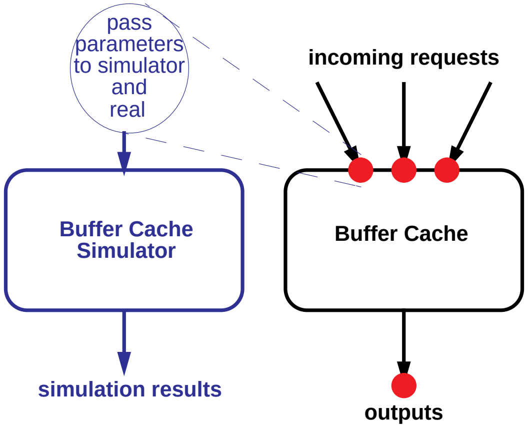
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

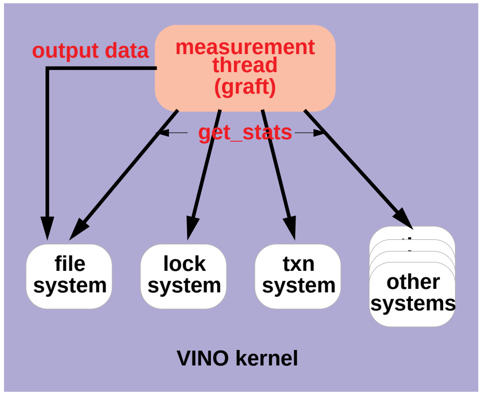
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

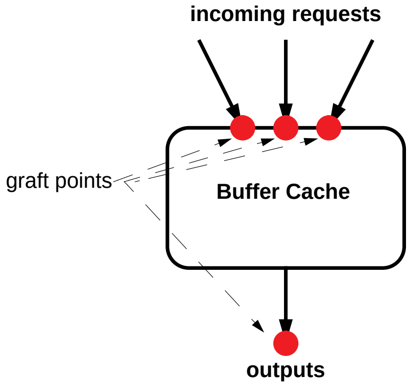
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

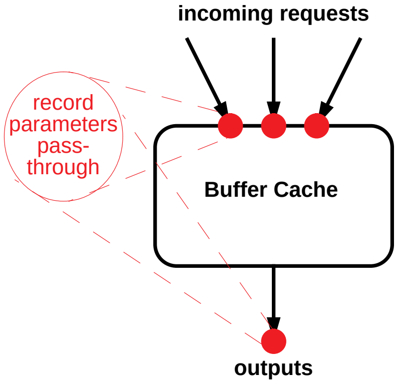
Measurement Thread



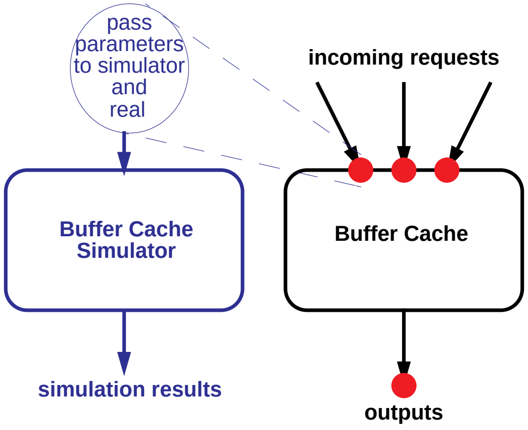
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

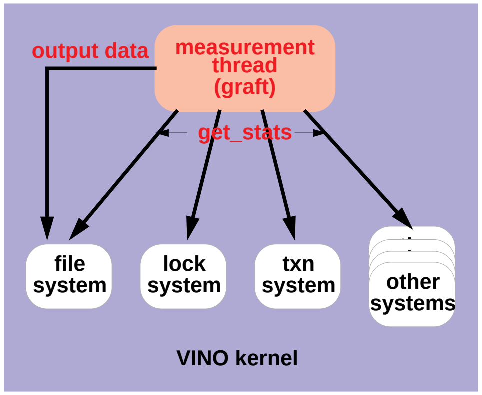
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

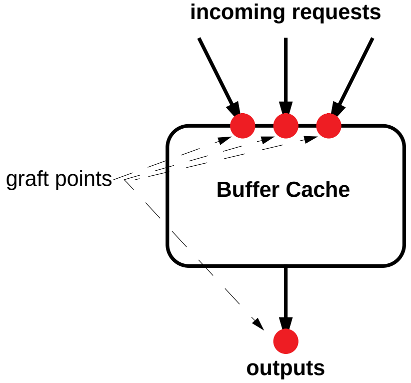
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

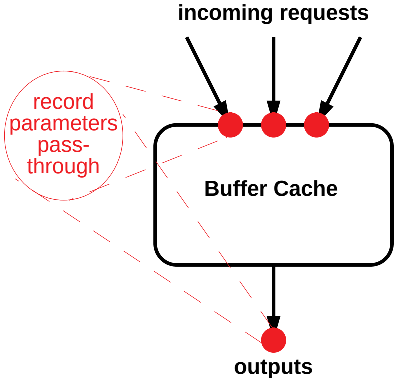
Measurement Thread


output data
measurement thread (graft)
get_stats
file system
lock system
txn system
other systems
VINO kernel

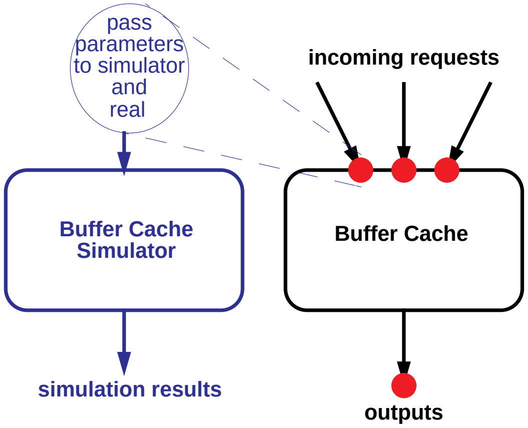
Generating Traces and Logs


incoming requests
graft points
Buffer Cache
outputs

Generating Traces and Logs


incoming requests
record parameters pass-through
Buffer Cache
outputs

In-Situ Simulation


pass parameters to simulator and real
incoming requests
Buffer Cache Simulator
Buffer Cache
simulation results
outputs

What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

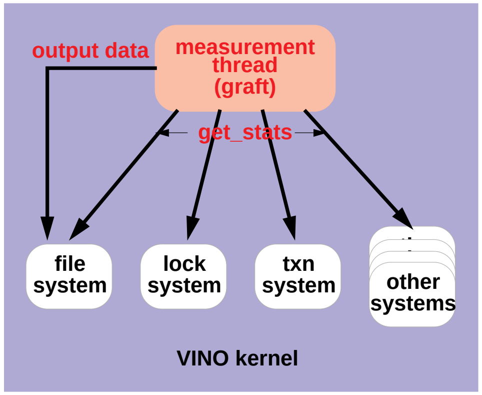
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

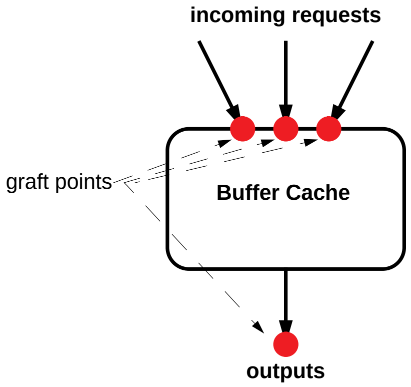
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

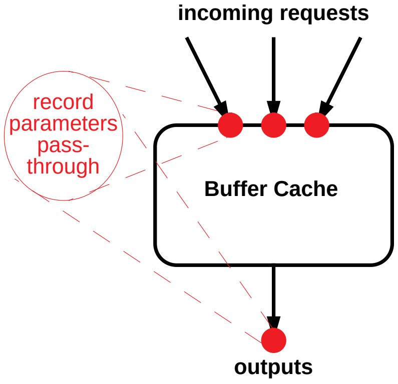
Measurement Thread



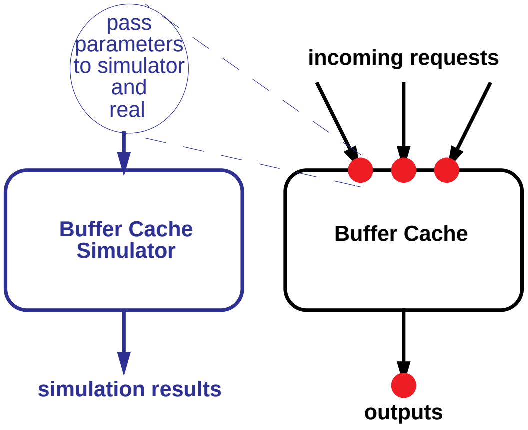
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

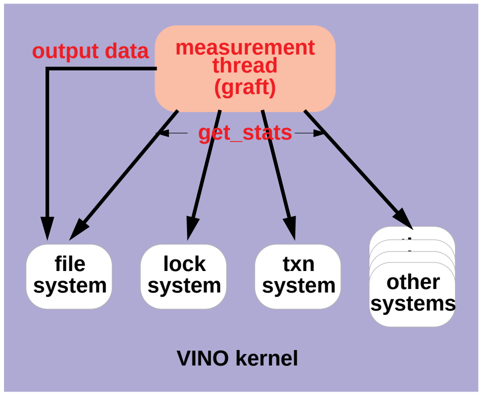
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

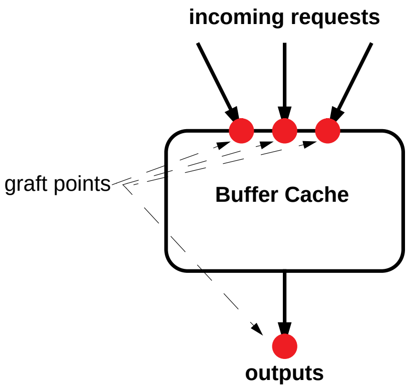
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

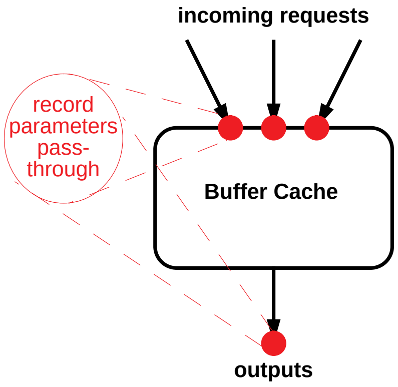
Measurement Thread



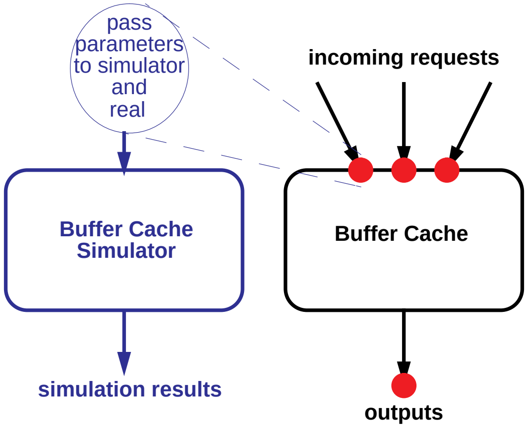
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

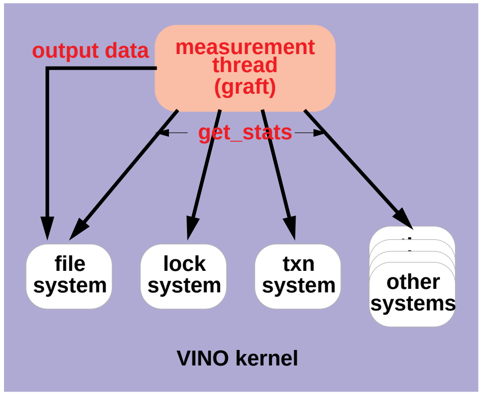
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

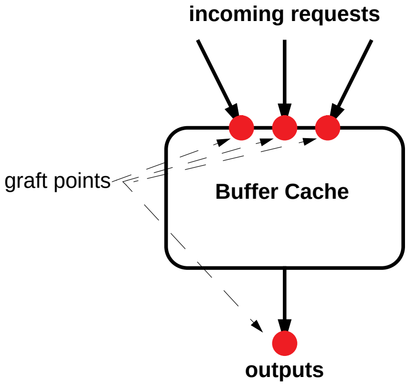
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

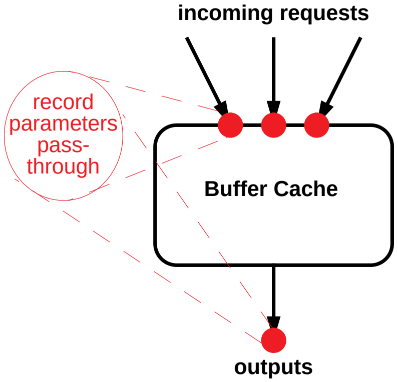
Measurement Thread



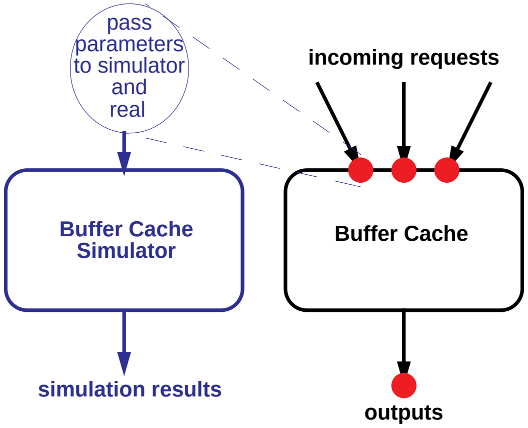
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

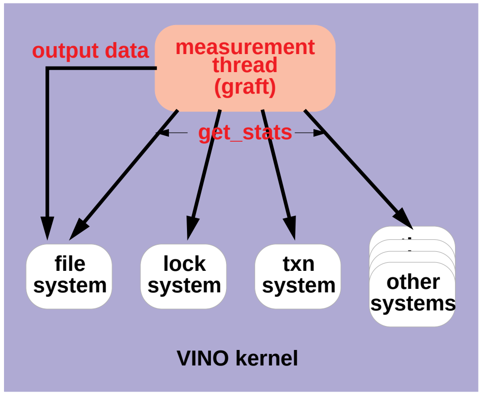
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

Measurement Thread



output data

measurement thread (graft)

get_stats

file system

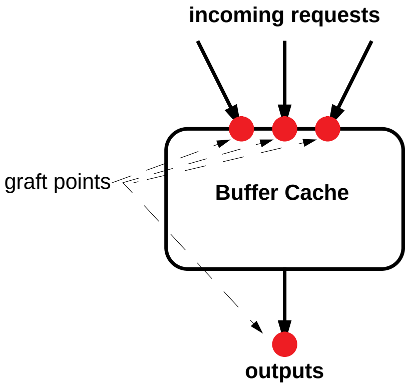
lock system

txn system

other systems

VINO kernel

Generating Traces and Logs



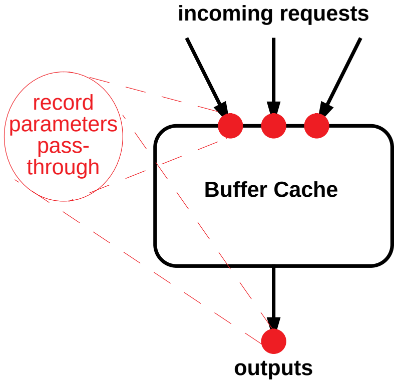
incoming requests

graft points

Buffer Cache

outputs

Generating Traces and Logs



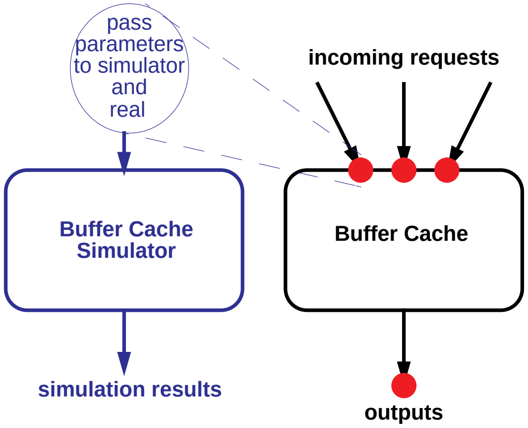
incoming requests

record parameters pass-through

Buffer Cache

outputs

In-Situ Simulation



pass parameters to simulator and real

incoming requests

Buffer Cache Simulator

Buffer Cache

simulation results

outputs

What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

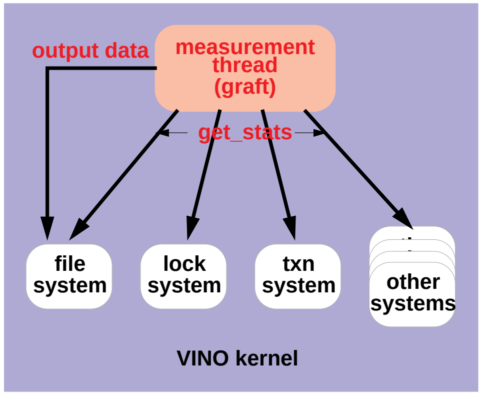
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

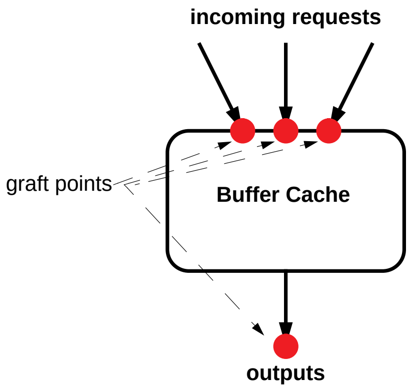
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

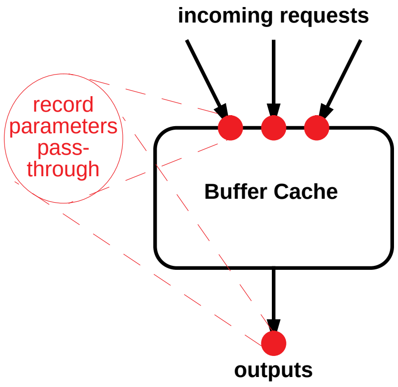
Measurement Thread


output data
measurement thread (graft)
get_stats
file system
lock system
txn system
other systems
VINO kernel

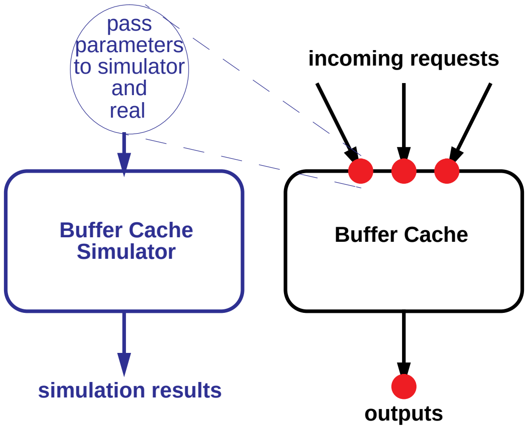
Generating Traces and Logs


incoming requests
graft points
Buffer Cache
outputs

Generating Traces and Logs


incoming requests
record parameters pass-through
Buffer Cache
outputs

In-Situ Simulation


pass parameters to simulator and real
incoming requests
Buffer Cache Simulator
Buffer Cache
simulation results
outputs

What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

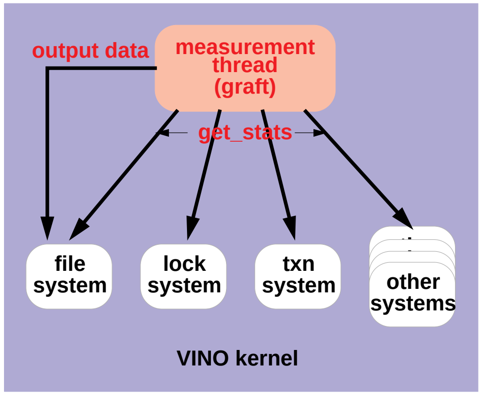
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

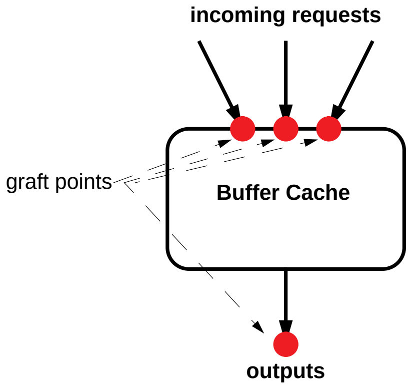
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

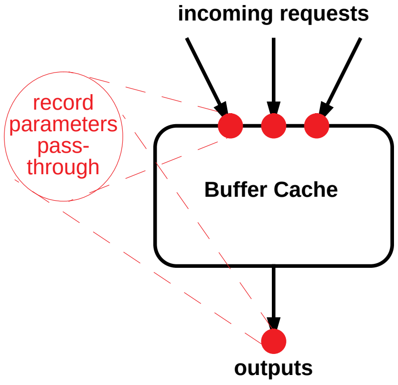
Measurement Thread



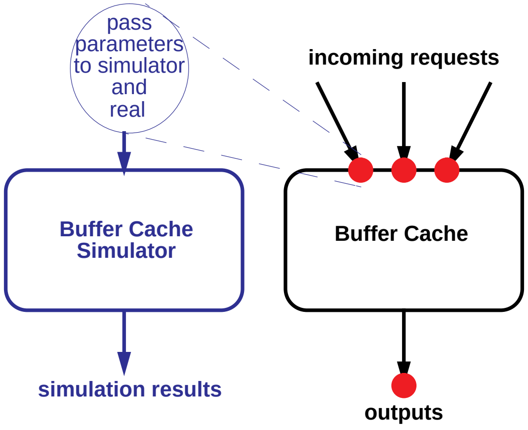
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

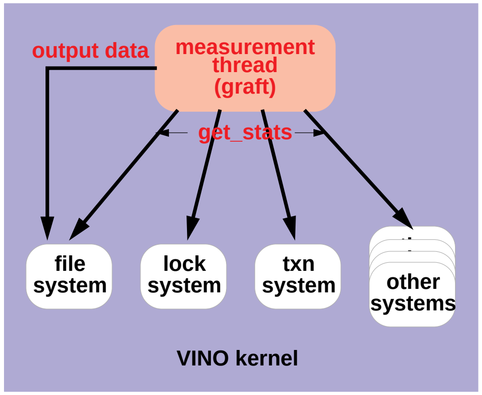
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

Measurement Thread



output data

measurement thread (graft)

get_stats

file system

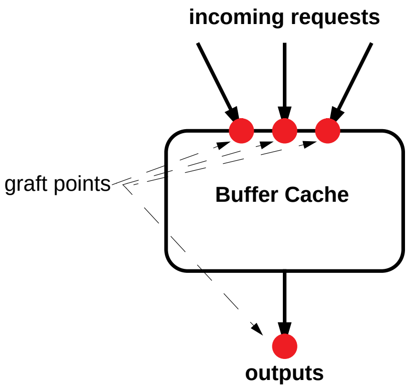
lock system

txn system

other systems

VINO kernel

Generating Traces and Logs



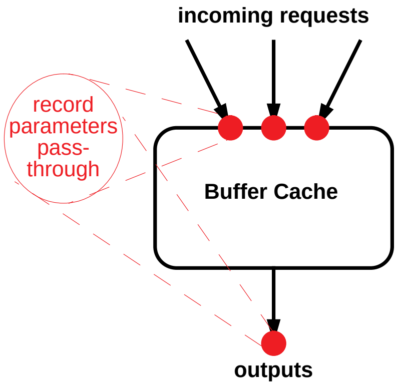
incoming requests

graft points

Buffer Cache

outputs

Generating Traces and Logs



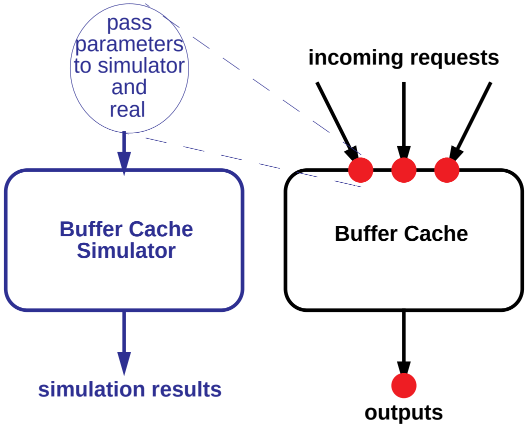
incoming requests

record parameters pass-through

Buffer Cache

outputs

In-Situ Simulation



pass parameters to simulator and real

incoming requests

Buffer Cache Simulator

Buffer Cache

simulation results

outputs

What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

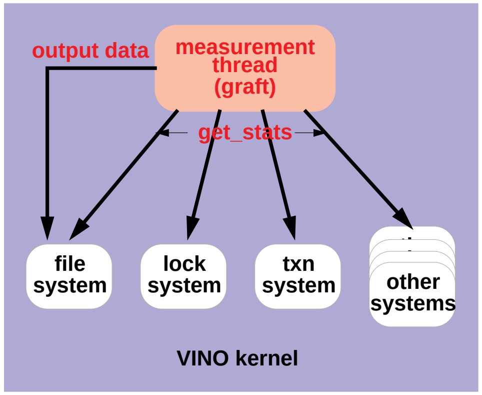
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

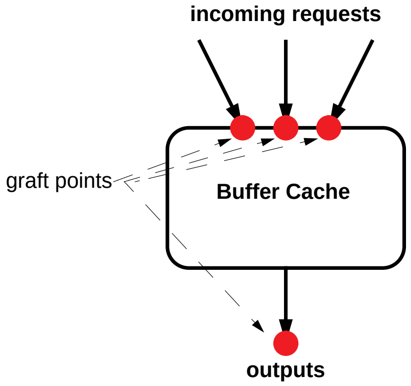
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

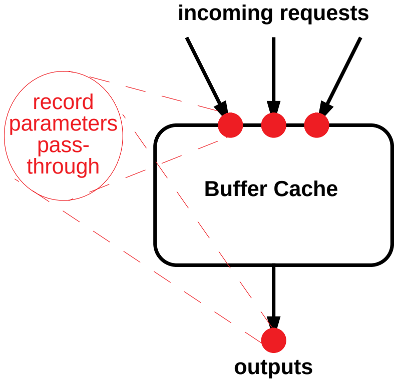
Measurement Thread



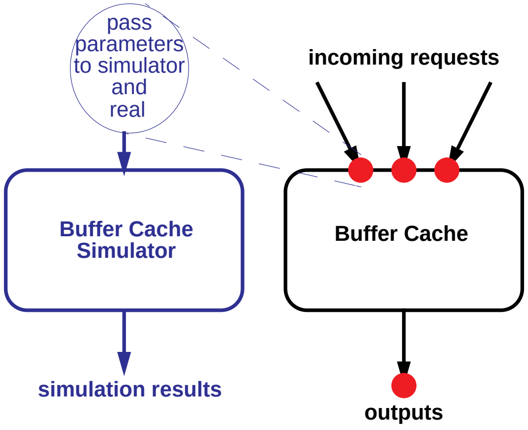
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

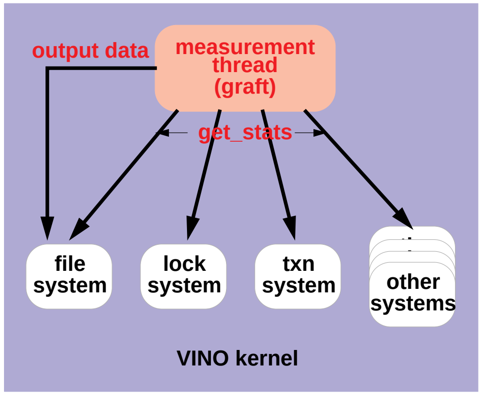
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

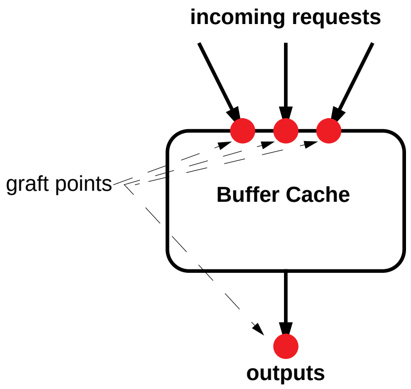
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

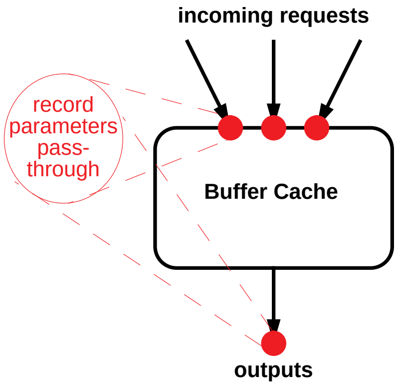
Measurement Thread



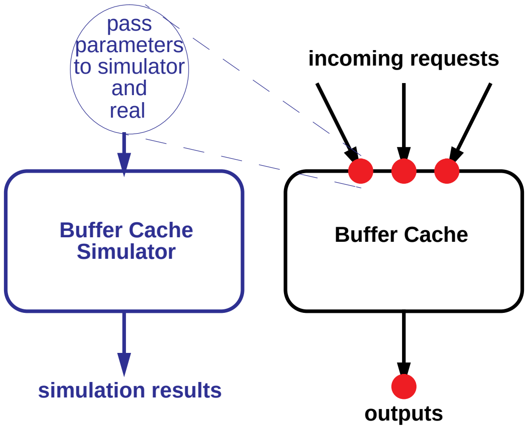
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

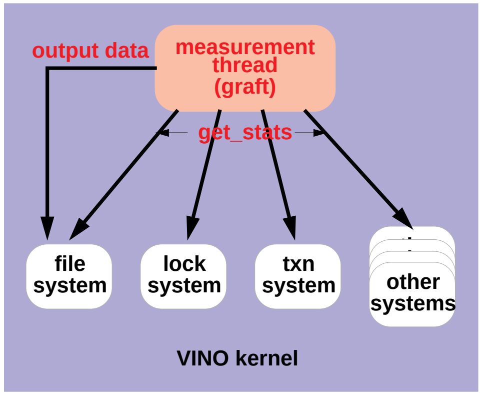
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

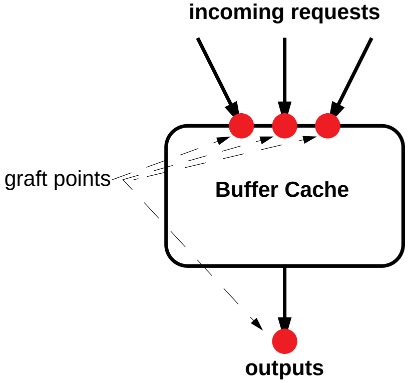
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

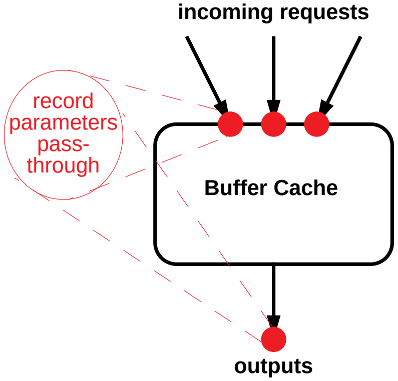
Measurement Thread



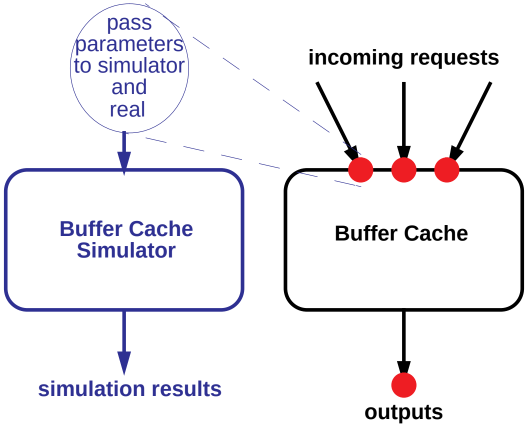
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

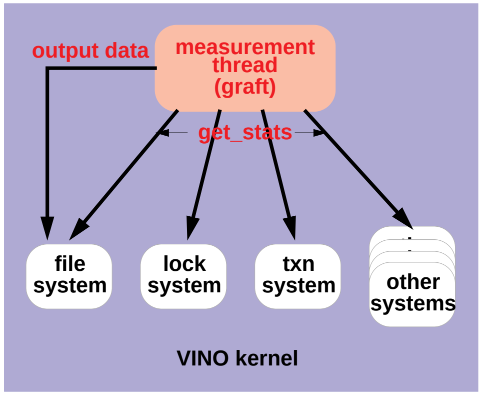
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

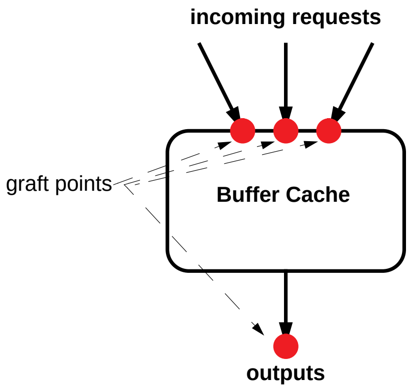
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

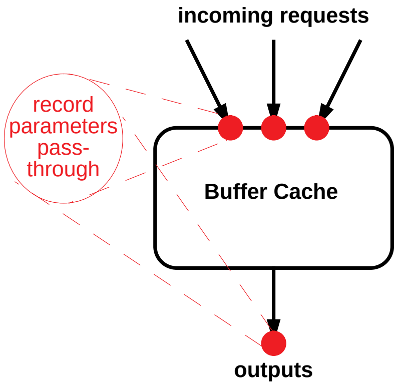
Measurement Thread



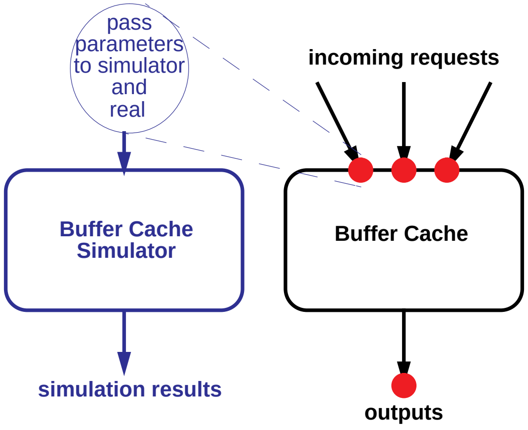
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

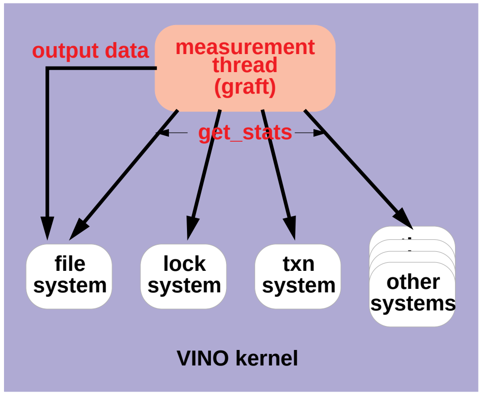
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

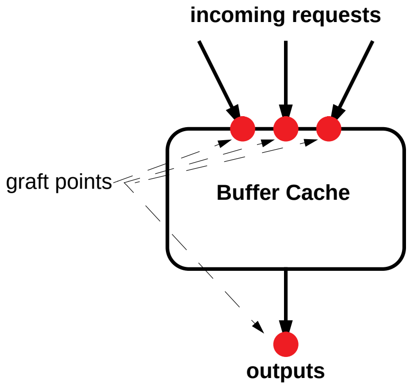
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

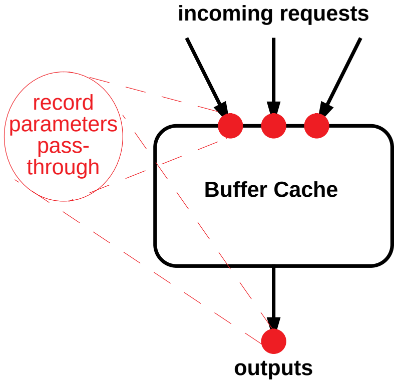
Measurement Thread



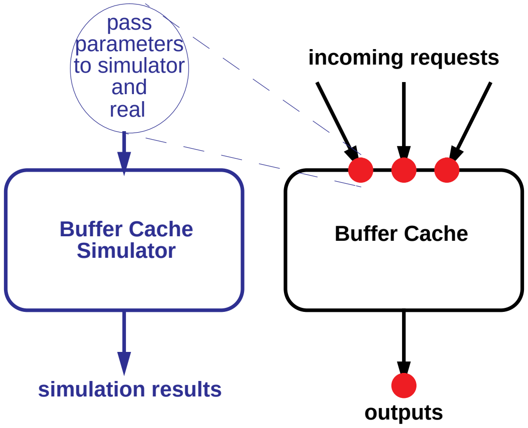
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- **Off-line Analysis**
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- **Online Analysis**
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

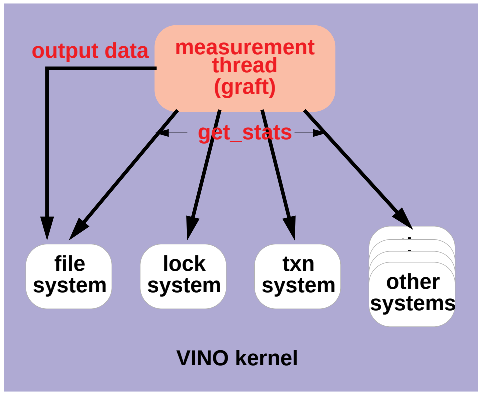
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

Measurement Thread



output data

measurement thread (graft)

get_stats

file system

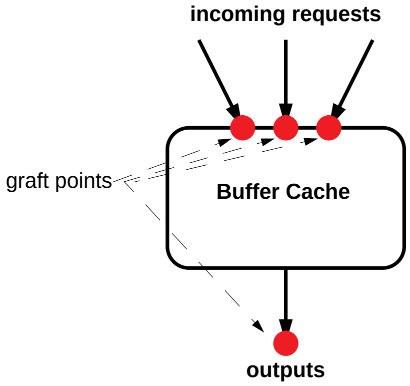
lock system

txn system

other systems

VINO kernel

Generating Traces and Logs



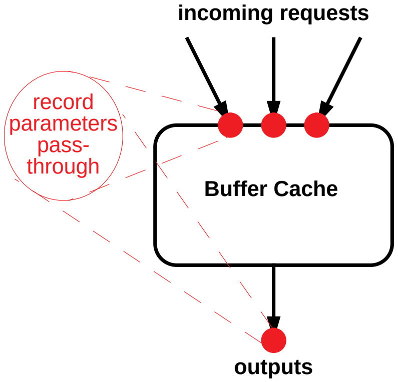
incoming requests

graft points

Buffer Cache

outputs

Generating Traces and Logs



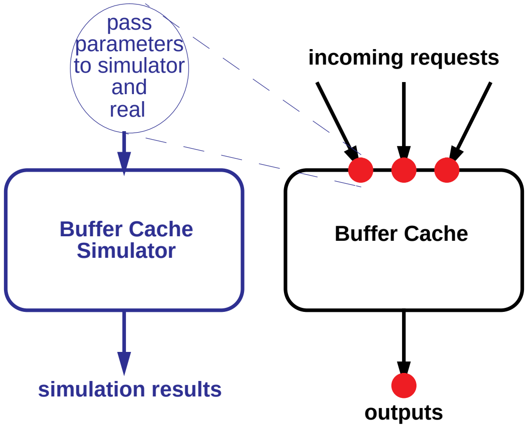
incoming requests

record parameters pass-through

Buffer Cache

outputs

In-Situ Simulation



pass parameters to simulator and real

incoming requests

Buffer Cache Simulator

Buffer Cache

simulation results

outputs

What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- **Use data from measurement thread to construct time series usage profile.**
- **Conduct variance analysis.**
- **Construct predicted usage profiles.**
- **Determine resource thresholds from predicted profiles.**
- **Notify online system of thresholds.**
- **Evaluate traces and logs; derive new algorithms.**
- **Simulate new algorithms, in situ.**

Online Analysis

- **Receive threshold and variance information from off-line system.**
- **Maintain dynamic statistics about:**
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- **Detect abnormal behavior.**
- **Dynamically trigger trace generation.**
- **Trigger adaptation heuristics.**

Adaptation Heuristics

- **Goal: decrease application latency.**
- **Paging**
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- **Disk Wait**
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- **CPU Hogs**
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**

Self-Monitoring and Self-Adapting Systems

Margo I. Seltzer

Harvard University

Division of Engineering and Applied Sciences

Assumptions

- Performance is important.
- People do not really know how to tune applications and systems.
- It would be nice to get some help from the system in tuning.
- Self-monitoring systems gather information about their own performance.

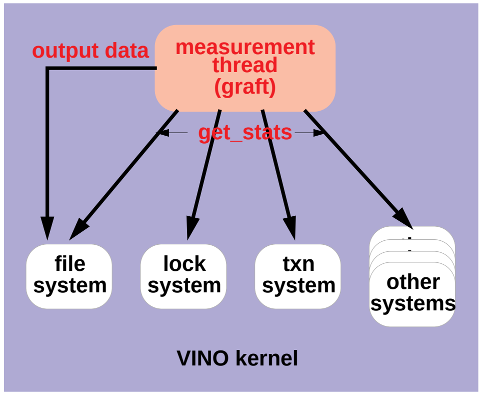
Outline

- Self-Monitoring in VINO.
- Processing monitor data.
- Adapting to system behavior.
- Conclusions.

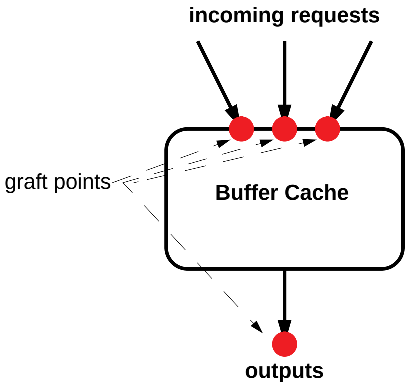
Self-Monitoring in VINO

- Measurement thread periodically collects module statistics.
- Generate detailed profiling information.
- Capture module inputs (*traces*) and outputs (*logs*).
- In-situ simulation evaluates competing algorithms and policies.

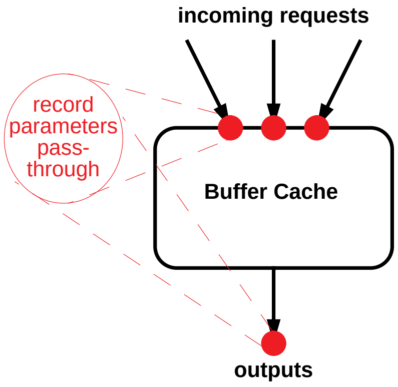
Measurement Thread



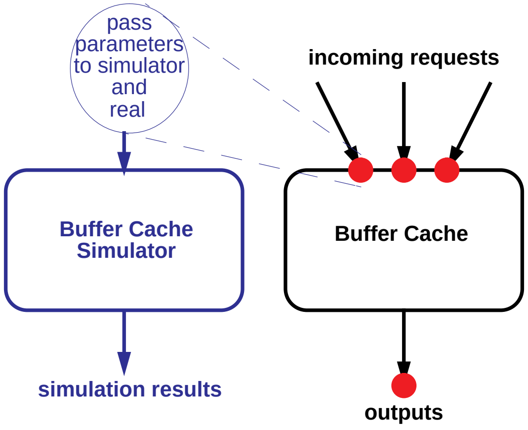
Generating Traces and Logs



Generating Traces and Logs



In-Situ Simulation



What do we do with Data?

- Off-line Analysis
  - Monitors long-term behavior.
  - Identifies common usage profiles.
  - Detects uncommon usage.
  - Suggests thresholds to online system.
  - Conducts feasibility evaluations.
- Online Analysis
  - Monitor instantaneous resource utilization.
  - Maintain efficiency statistics.
  - Detect dangerous conditions.

Off-line Analysis

- Use data from measurement thread to construct time series usage profile.
- Conduct variance analysis.
- Construct predicted usage profiles.
- Determine resource thresholds from predicted profiles.
- Notify online system of thresholds.
- Evaluate traces and logs; derive new algorithms.
- Simulate new algorithms, in situ.

Online Analysis

- Receive threshold and variance information from off-line system.
- Maintain dynamic statistics about:
  - Cache hit rates.
  - Lock contention.
  - Disk queue lengths.
  - Load averages.
  - Context switch rates.
- Detect abnormal behavior.
- Dynamically trigger trace generation.
- Trigger adaptation heuristics.

Adaptation Heuristics

- Goal: decrease application latency.
- Paging
  - Collect page access trace.
  - Look for well-known patterns (linear, cyclic, strided).
  - Look for page access correlation.
  - Install better prefetching algorithm.
- Disk Wait
  - Similar process to paging.
  - Replace read-ahead for the application(s).
- CPU Hogs
  - Examine profile output.
  - Recompile kernel modules in application context.

Adaptation (continued)

- **Interrupt Latency**
  - Measure latency between interrupt arrival and delivery to process/thread.
  - Look for excessively long intervals or high variance.
  - Check (fix) scheduling priorities.
- **Lock Contention**
  - Measure lock wait times.
  - Decrease lock granularity on highly contested items.

Conclusions

- **Self-monitoring is a generally useful idea.**
- **An extensible system just makes it easier.**
- **Automatic adaptation is a cool idea.**
- **Challenging to do it correctly.**
- **An extensible system makes it easier to experiment with this.**