Treadmill: Attributing the Source of Tail Latency through Precise Load Testing and Statistical Inference

Yunqi Zhang, David Meisner, Jason Mars, Lingjia Tang
Internet services

- User interactive applications
- Powered by large-scale distributed systems
- Millions of queries hitting the servers
Tail latency

- Orders of magnitude higher than average latency
- Negatively affects user experience
- Resource provisioned based on tail latency
It is challenging for service providers to keep the tail of latency distribution short for interactive services as the size and complexity of the system scales up.

— Jeffrey Dean, Luiz Barroso
“The Tail at Scale”
Google
Challenges

• Tail latency is sensitive to any variance in the system

• Many services operate at latency as low as microseconds

• Many architectural components are involved
Limitations of prior studies

• **NUMA** [NUMA Experience 'HPCA 2013, Tales of the Tail 'SoCC 2014]

• **NIC** [Architecting to Achieve a Billion Request per Second Throughput 'ISCA 2015, Tales of the Tail 'SoCC 2014, Chronos 'SoCC 2012]


Architectural components have **complex interactions**
Goal

Attribute and understand the source of tail latency
Tail latency measurement

**CloudSuite [ASPLOS 2012]**

**Mutilate [EuroSys 2014]**

Why are they different?
Tail latency measurement

- Client-side queueing bias
- Query inter-arrival generation
- Statistical aggregation
- Performance hysteresis
Client-side queueing bias

Treadmill: Attributing the Source of Tail Latency through Precise Load Testing and Statistical Inference
Client-side queueing bias

Multiple clients are needed to avoid client-side queueing bias

Treadmill: Attributing the Source of Tail Latency through Precise Load Testing and Statistical Inference
Query inter-arrival generation

[Closed v.s. Open System Models. ‘NSDI 2006]

closed-loop

open-loop

[Image showing closed-loop and open-loop models with graphs and data points]

CDF

Number of Outstanding Requests

Open-Loop
Closed-Loop w/12 Connections
Closed-Loop w/8 Connections
Closed-Loop w/4 Connections
Query inter-arrival generation

Open-loop is necessary to properly exercise the system queueing behavior

[Closed v.s. Open System Models. ‘NSDI 2006]
Treadmill

• Open source
  • [https://github.com/facebook/treadmill](https://github.com/facebook/treadmill)

• Generality
  • < 200 lines of code to integrate each workload

• CloudSuite [ASPLOS2012]
• Mutilate [EuroSys2014]
• Treadmill
Evaluation

- Extremely long tail
- Client-side queueing bias
- Slightly long tail
- Different distribution
- Exact same shape
- Fixed gap to tcpdump
Evaluation

- Extremely long tail
- Client-side queueing bias

Treadmill achieves **microsecond-level precision** even at high quantiles

- Exact same shape
- Fixed gap to tcpdump
Goal

Attribute and understand the source of tail latency
Tail latency attribution

• Quantile regression
  
  (tail latency) (architectural components)
  • attribute variance to explanatory variables and their interactions
  • works with any given quantile rather than average only (ANOVA)

• Architectural components
  
  • NUMA allocation policy
  • DVFS frequency governor
  • TurboBoost
  • NIC interrupt handling
Complex interactions

- NIC IRQ policy
  - same-node
  - all-nodes
- DVFS Governor
  - ondemand
  - performance

NIC IRQ same-node

more frequency transition overhead

NIC IRQ all-nodes

Utilization
Complex interactions

- NIC IRQ policy
  - same-node
  - all-nodes
- DVFS Governor
  - ondemand
  - performance

NIC IRQ same-node

NIC IRQ all-nodes

more frequency transition overhead

Tail latency attribution enables understanding of complex interactions
Tail latency reduction

43% reduction in 99th-percentile latency
93% reduction in its variance
Conclusion

• Identifying common pitfalls
  • query inter-arrival generation; client-side queueing bias; statistical aggregation; performance hysteresis

• Treadmill
  • open source modular load testing infrastructure
  • achieves high precision at microsecond-level

• Attributing the source of tail latency
  • understand complex interactions among architectural components
  • 43% reduction in 99th-percentile latency
Thank you!

https://github.com/facebook/treadmill