TOWARD PREDICTABLE PERFORMANCE IN SOFTWARE PACKET-PROCESSING PLATFORMS

Mihai Dobrescu, EPFL
Katerina Argyraki, EPFL
Sylvia Ratnasamy, UC Berkeley

Programmable Networks

- Industry/research community efforts
  - Easily deploy new services
  - Test research ideas
- Software packet processing
  - General purpose hardware
  - Familiar programming environment

Extensible network functionality

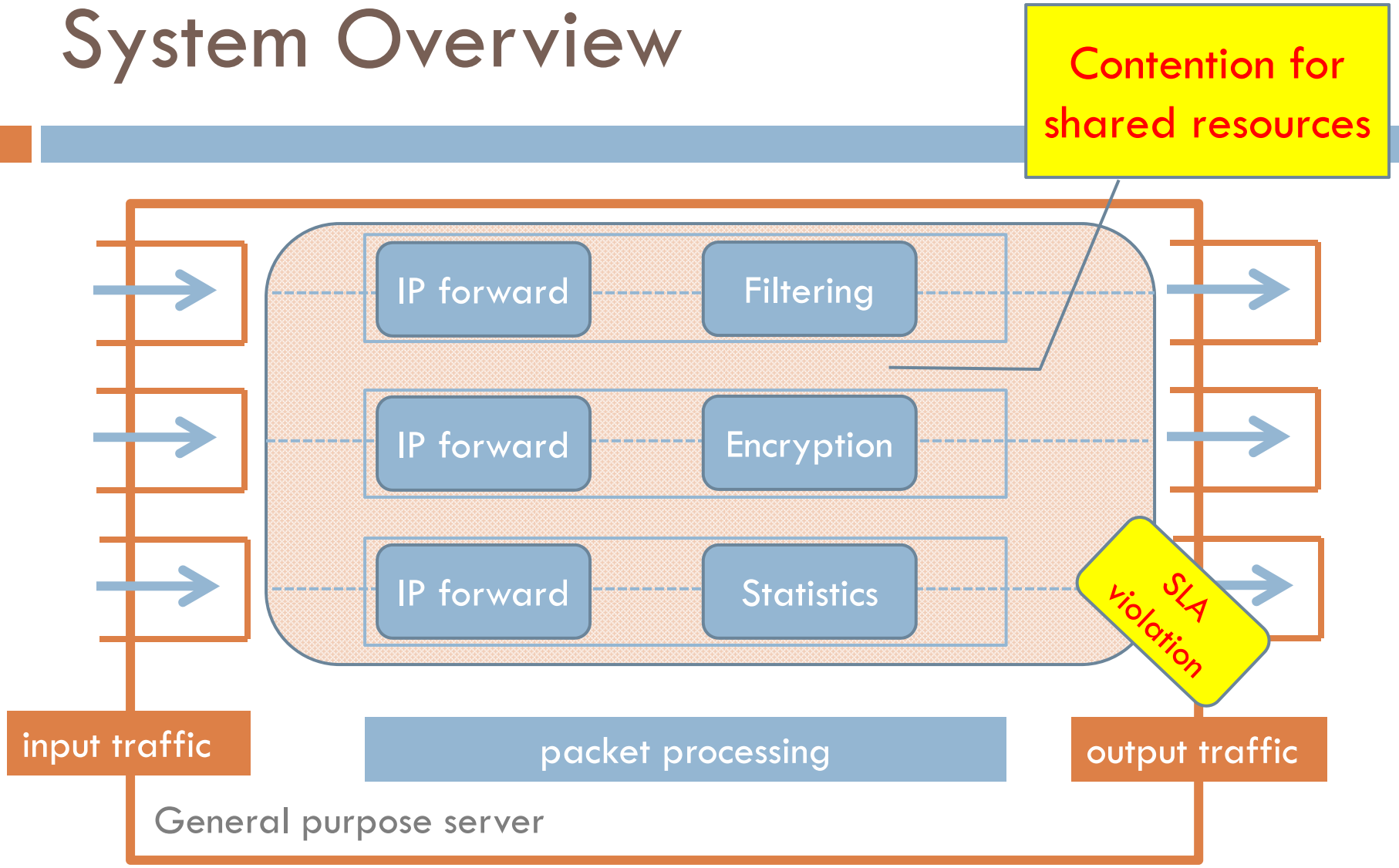
Problem: Unpredictable Performance

- Resource contention
  - Caches, memory controllers, buses
  - Performance interference

- Software packet-processing systems [Dobrescu'09, Han'10]
  - High performance
  - Same processing for all packets

Goal: software packet processing with predictable performance

System Overview



Is This Hard?

- Yes, in general-purpose context
  - Math models to predict contention
  - Contention-aware job placement
- In packet-processing context?

Our Contribution

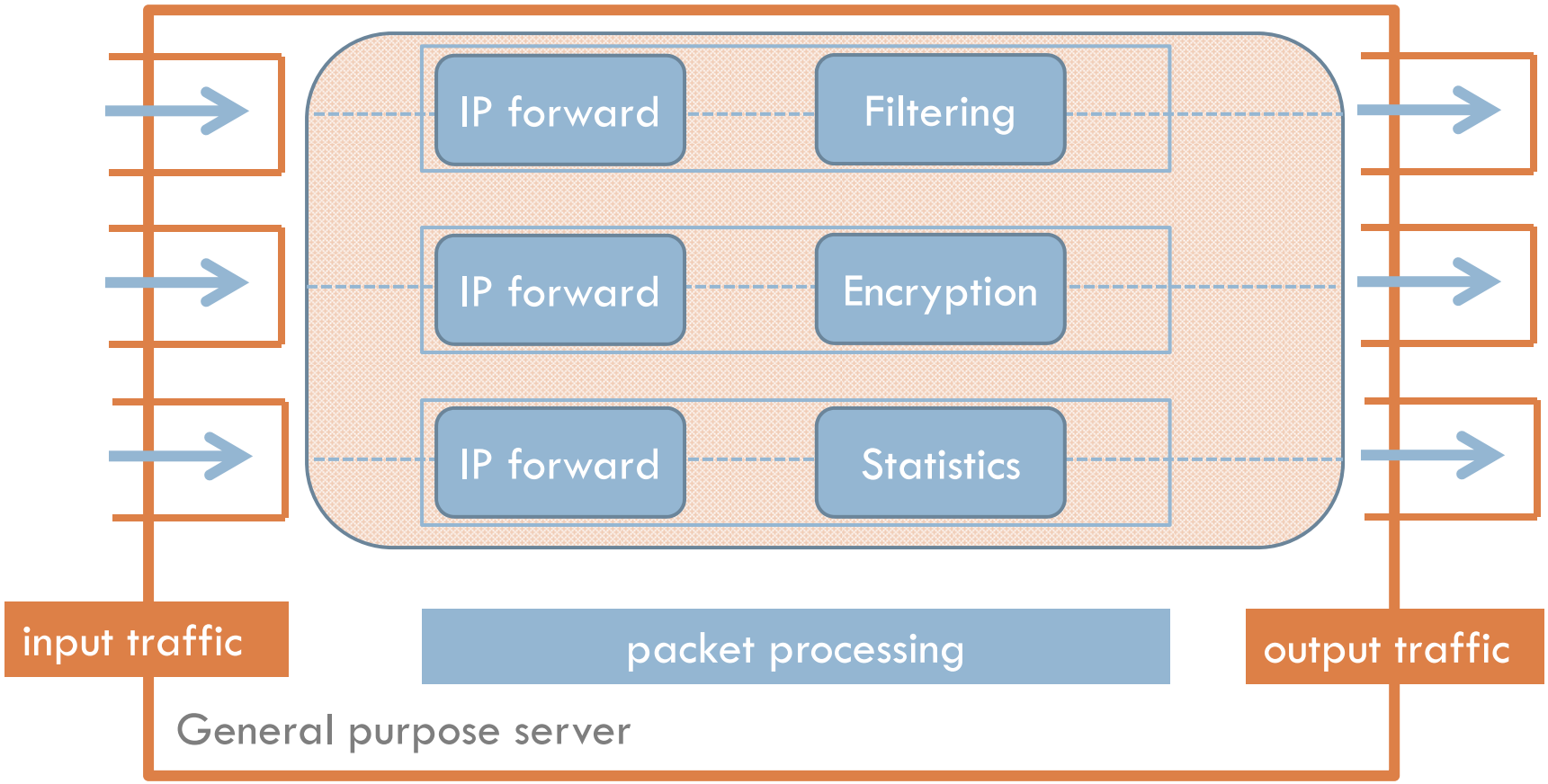
1. It is feasible to build a packet-processing platform with predictable performance using simple techniques.

2. Contention-aware job placement does not bring significant benefit to the overall performance.

Outline

- System overview
- Contention factors
- Observations on application behavior
- A simple prediction method
- Intuition

System Overview



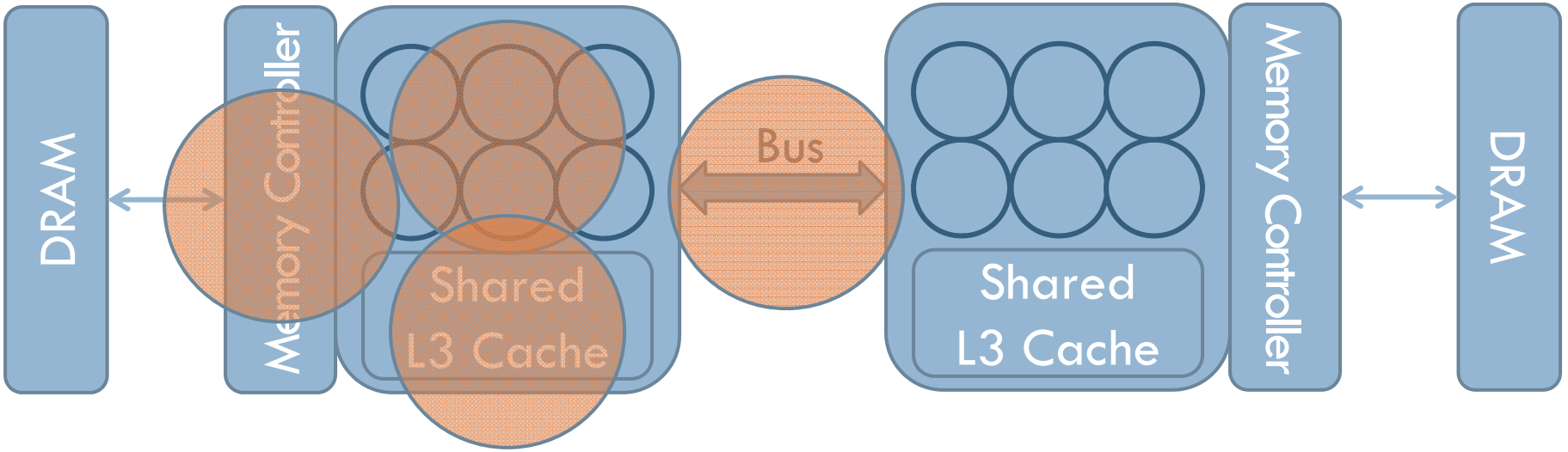
Workloads

Application	Main functionality	Characteristics
IP	IP routing, 128k entries	L3 cache intensive
MON	Monitoring, 100k flows	L3 cache intensive
FW	Firewall, 1000 rules	L2 cache intensive
RE	Redundancy elimination	Memory intensive
VPN	Encryption	CPU intensive
Synthetic	Random cache reads	Cache/memory/CPU

Representative set of realistic applications

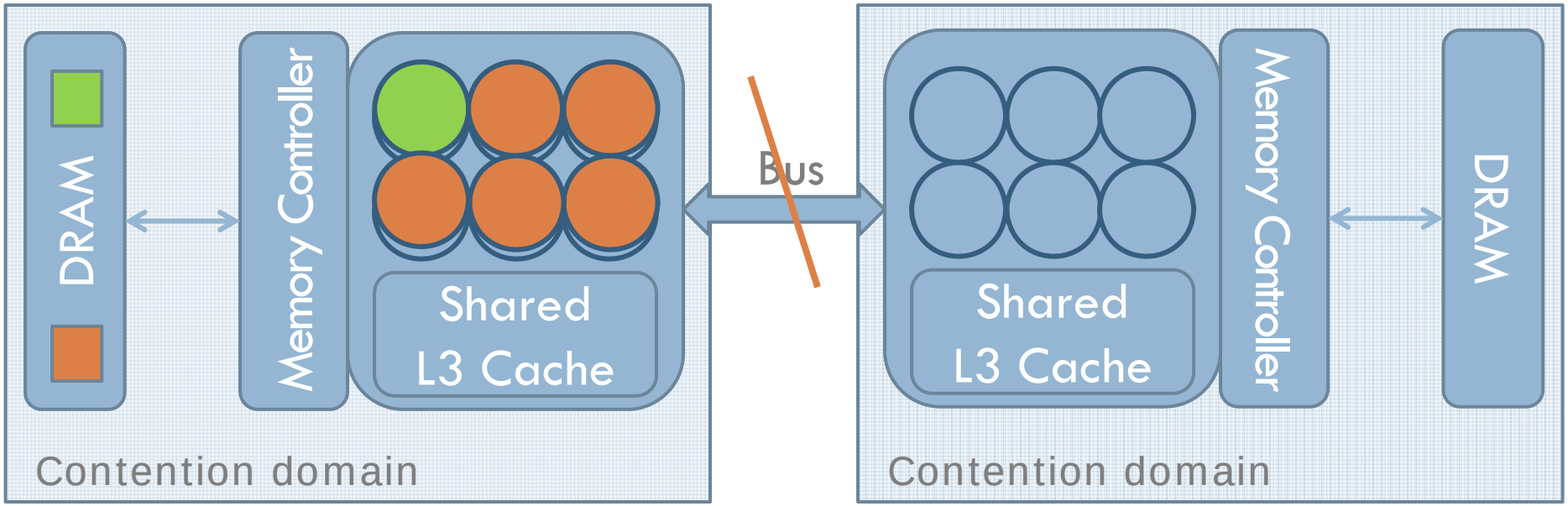
Setup

- Linux + Click
- Commodity Intel Xeon server



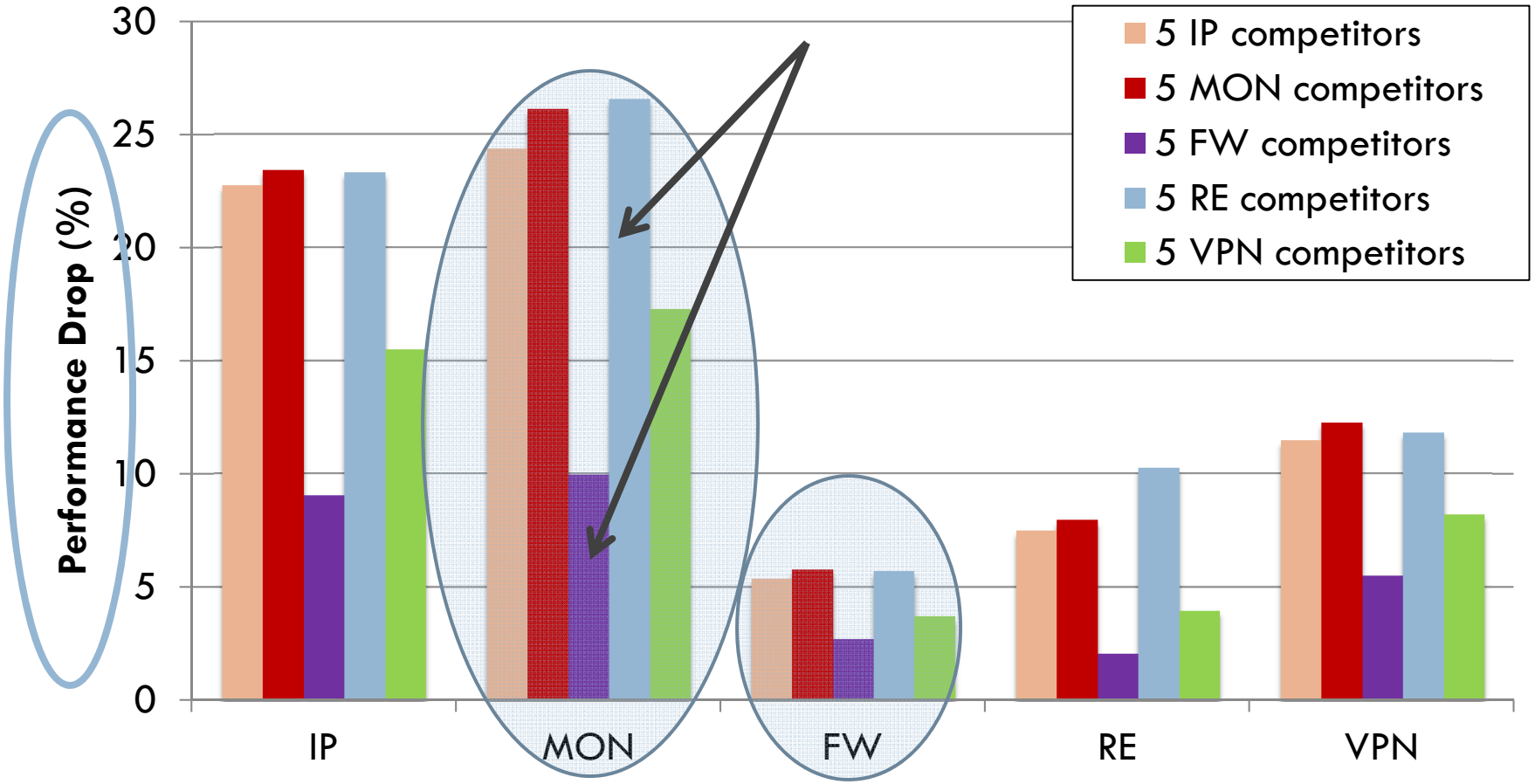
Basic Configuration

- One application per core
- NUMA-aware memory allocation



Contended resources: cache *and* memory controller

Resource Contention Effects

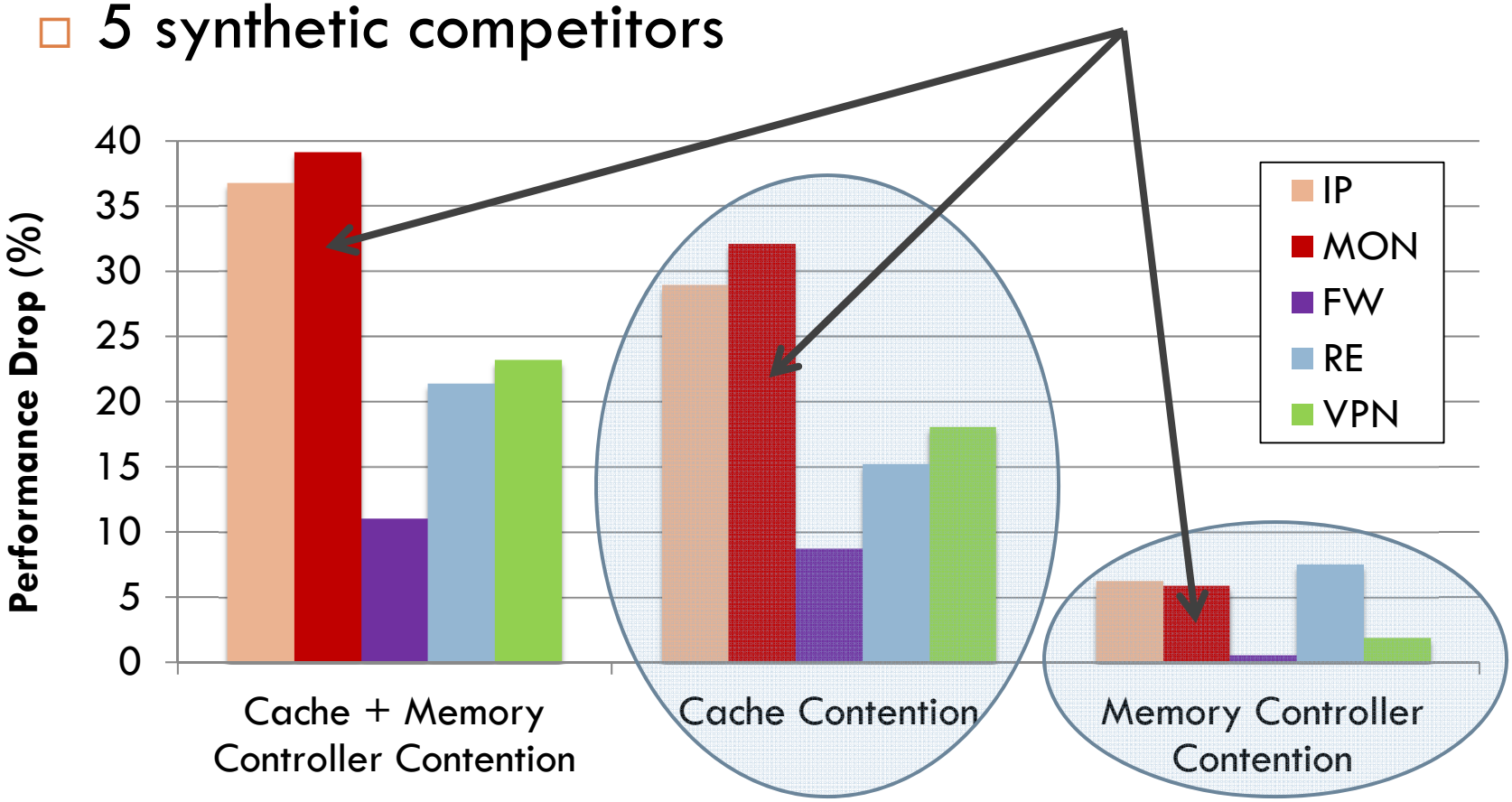


Outline

- System overview
- Contention factors
- Observations on application behavior
- A simple prediction method
- Intuition

Contention Factors

- 5 synthetic competitors

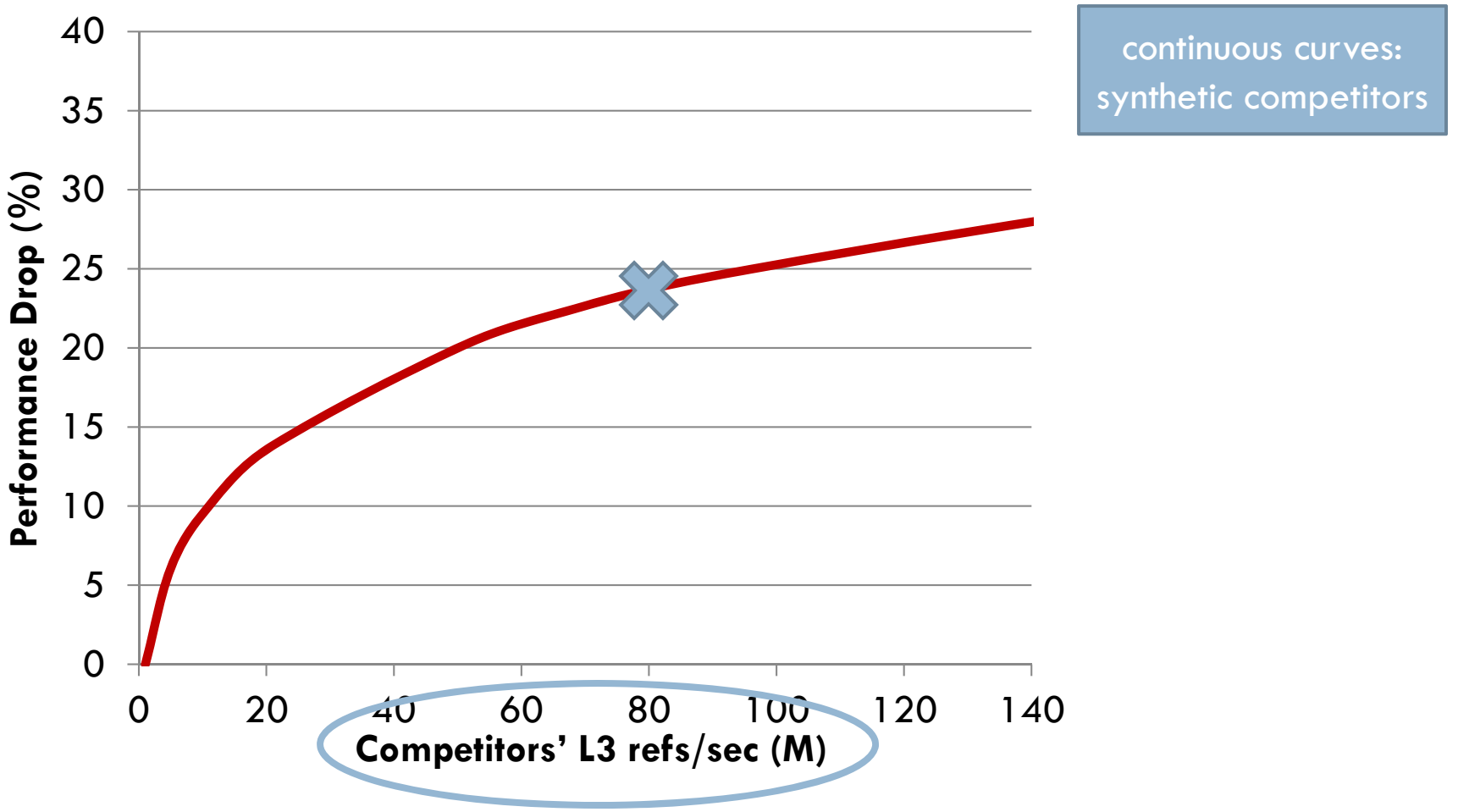


Cache is the dominant contention factor

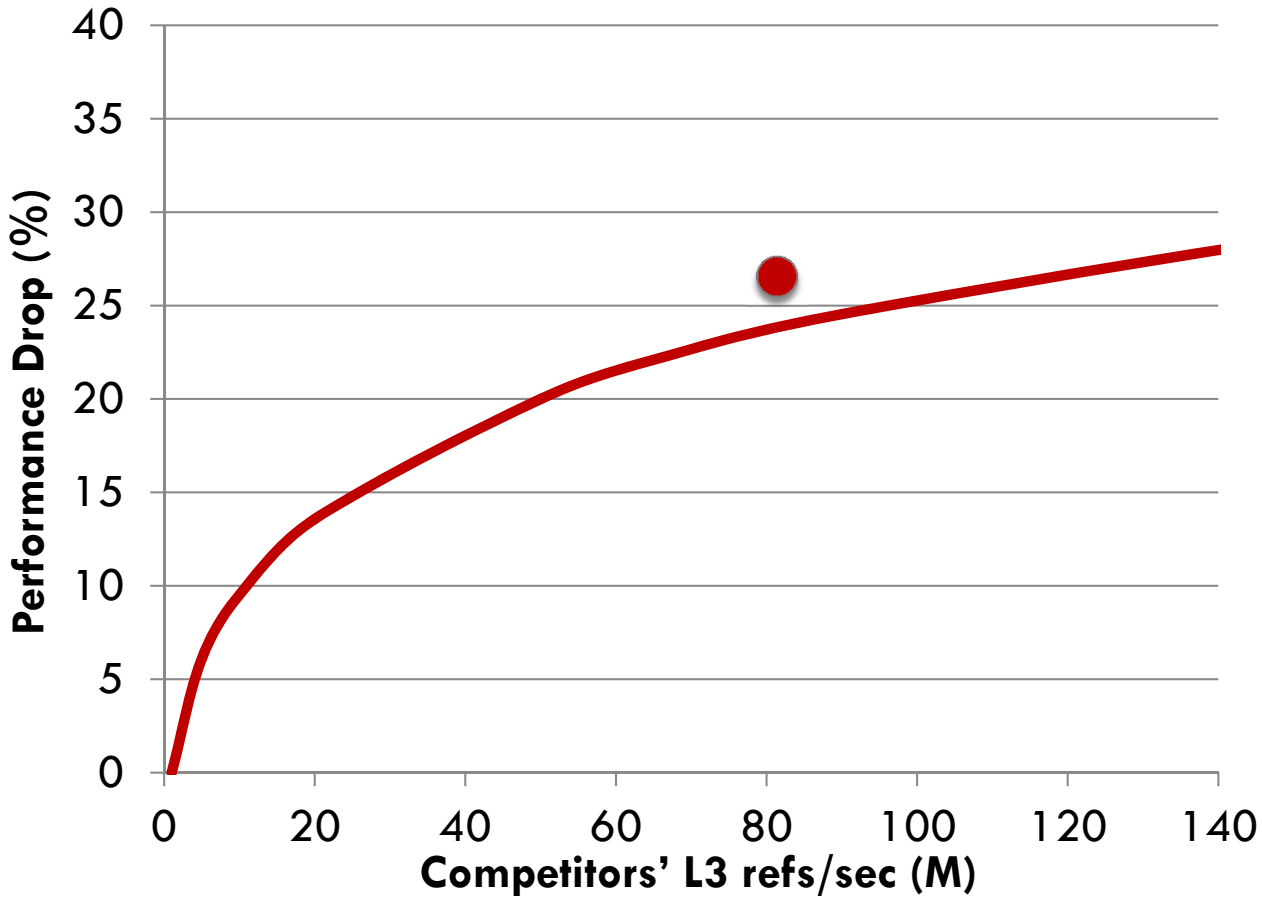
Outline

- System overview
- Contention factors
- Observations on application behavior
- A simple prediction method
- Intuition

Characterize Application Behavior



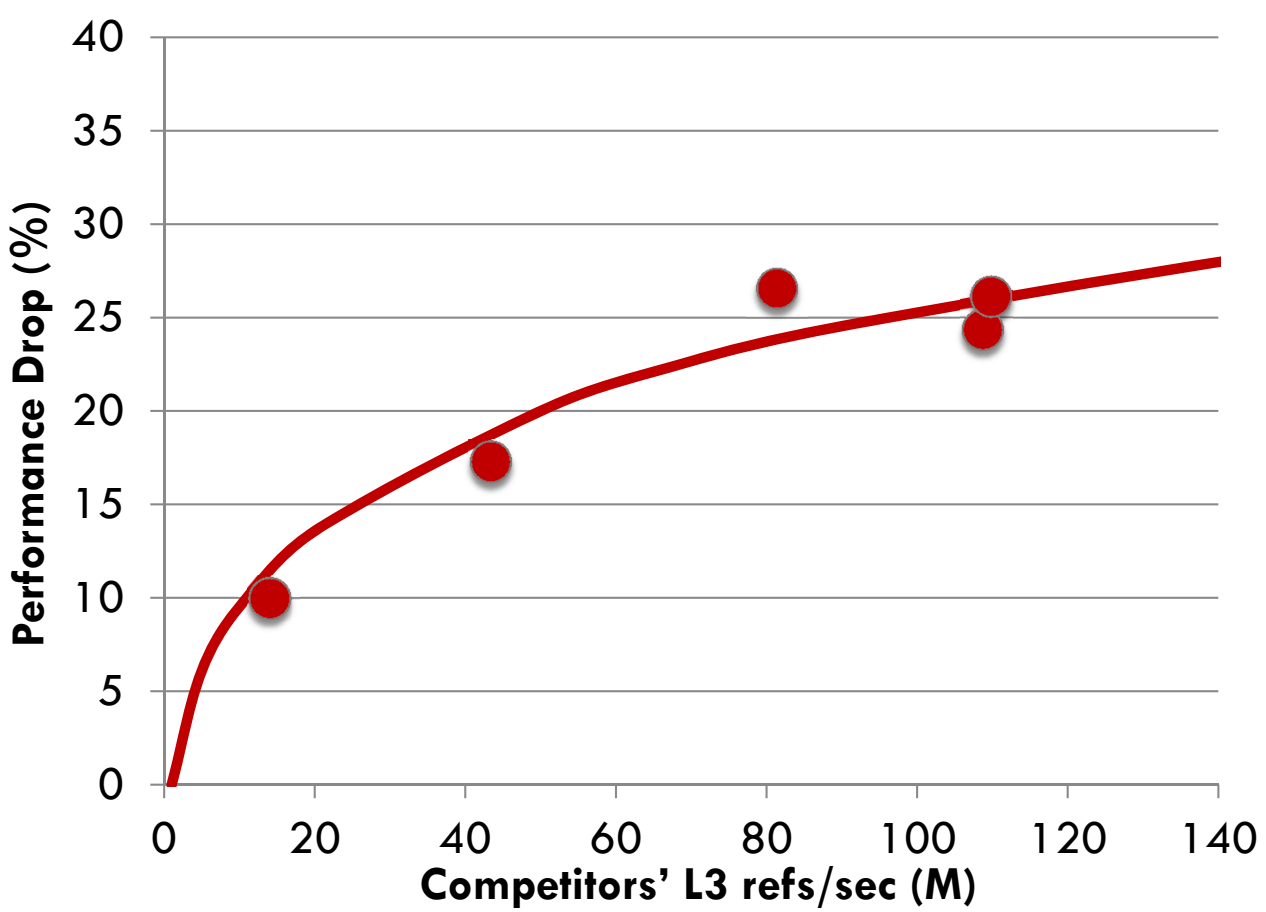
Characterize Application Behavior



continuous curves: synthetic competitors

individual points: realistic competitors

Characterize Application Behavior

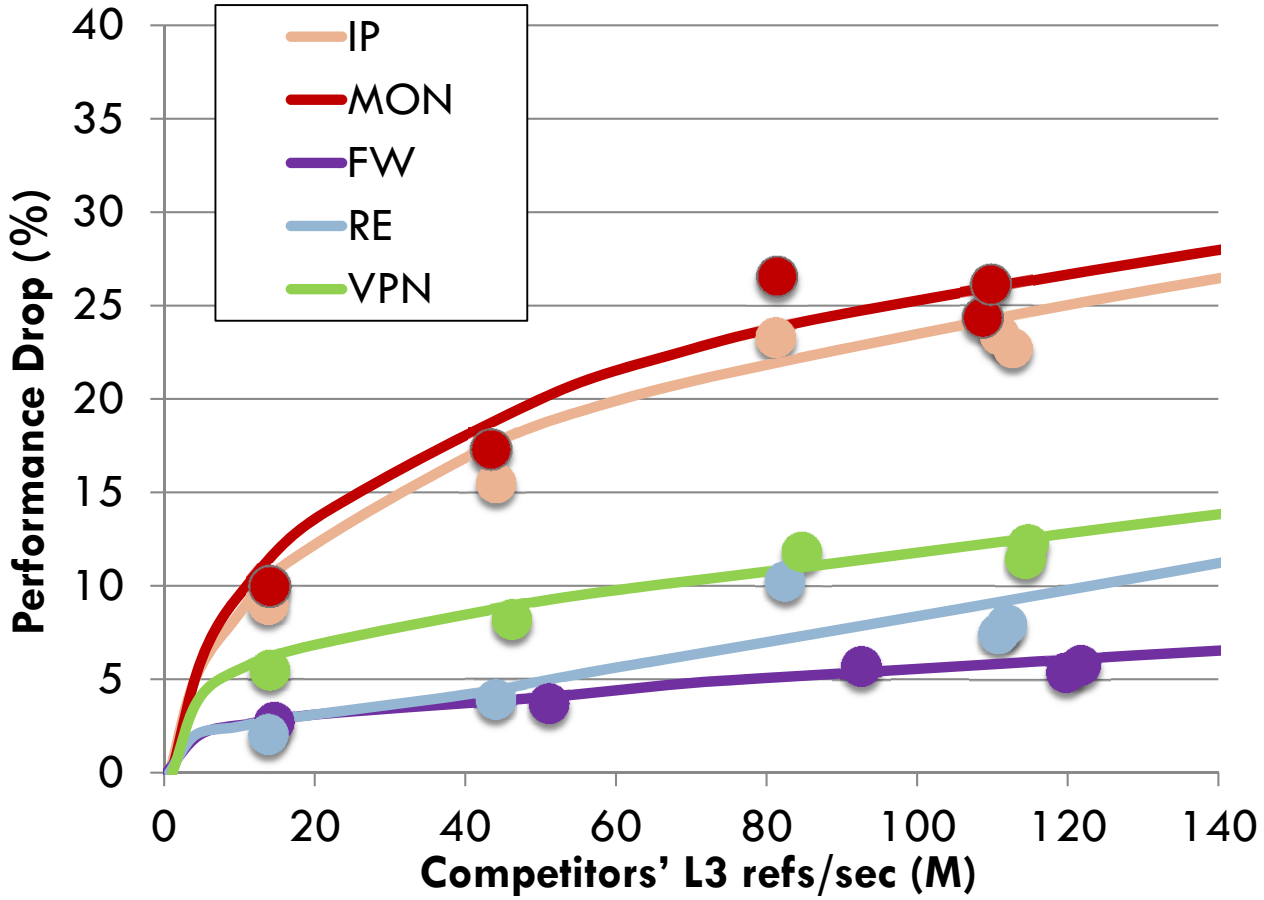


continuous curves: synthetic competitors

individual points: realistic competitors

Obs. #1: competitors' cache refs/sec determine drop

Characterize Application Behavior

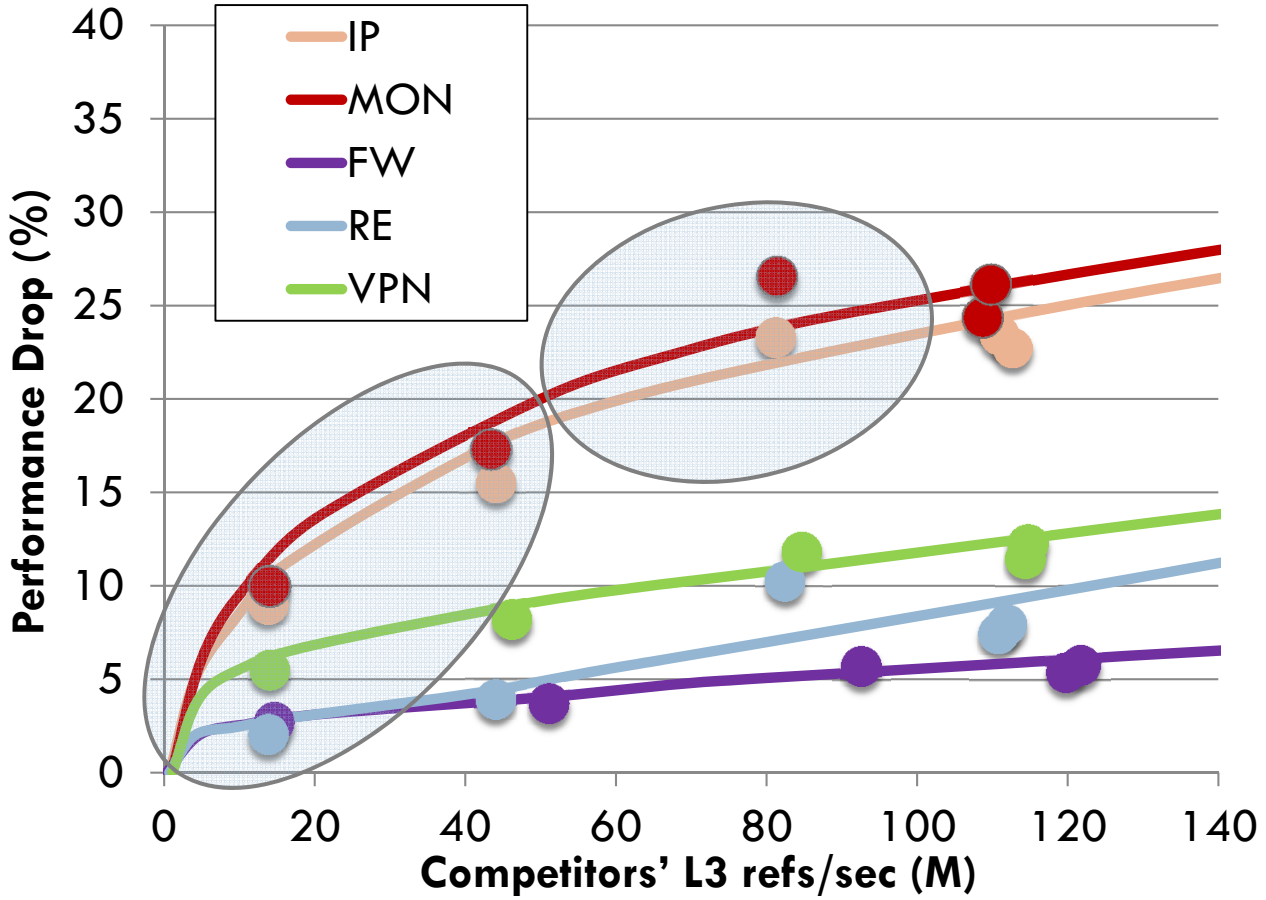


continuous curves: synthetic competitors

individual points: realistic competitors

Obs. #1: competitors' cache refs/sec determine drop

Characterize Application Behavior



continuous curves: synthetic competitors

individual points: realistic competitors

Obs. #2: drop curve grows slowly after certain point

Outline

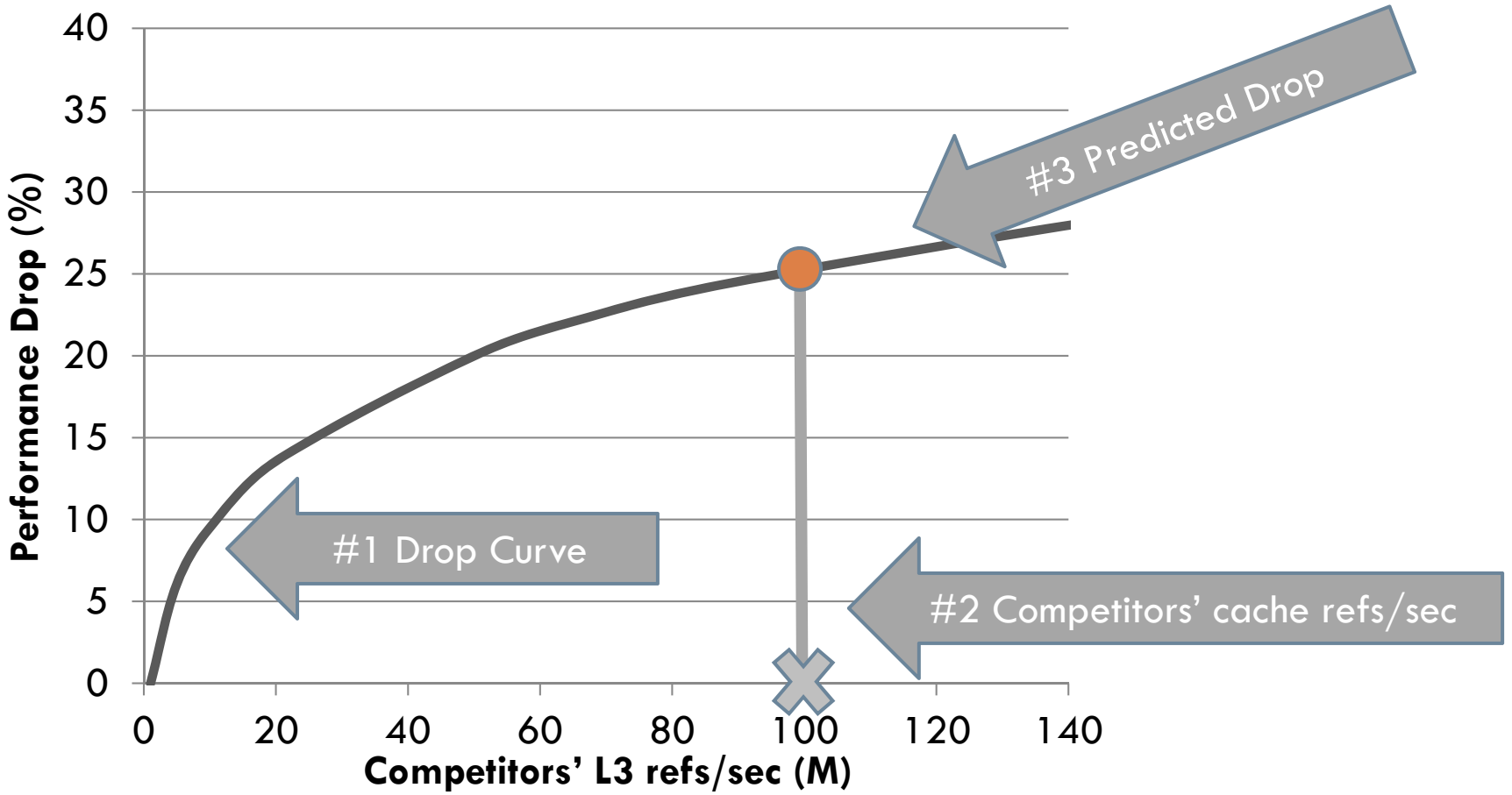
- System overview
- Contention factors
- Observations on application behavior
- A simple prediction method
- Intuition

Contention Effects Prediction

- Step#1: performance drop curve for each app
  - Synthetic competitors – random cache reads
  - Vary competitors' cache refs/sec
- Step#2: cache refs/sec for each app running alone
- Step#3: predicted drop equals the value of the drop curve corresponding to the competing cache refs/sec

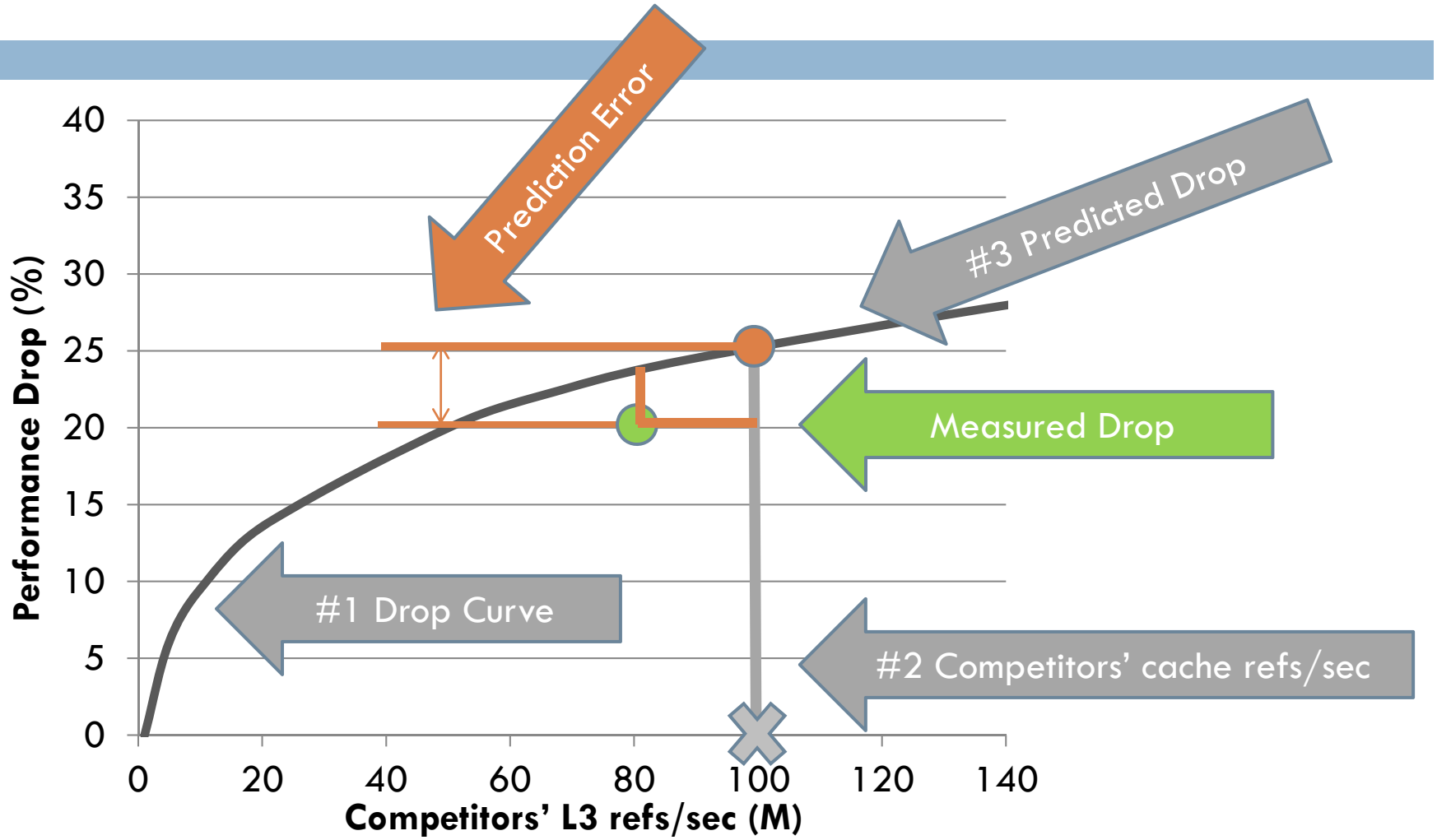
Simple offline profiling

Step by Step Prediction

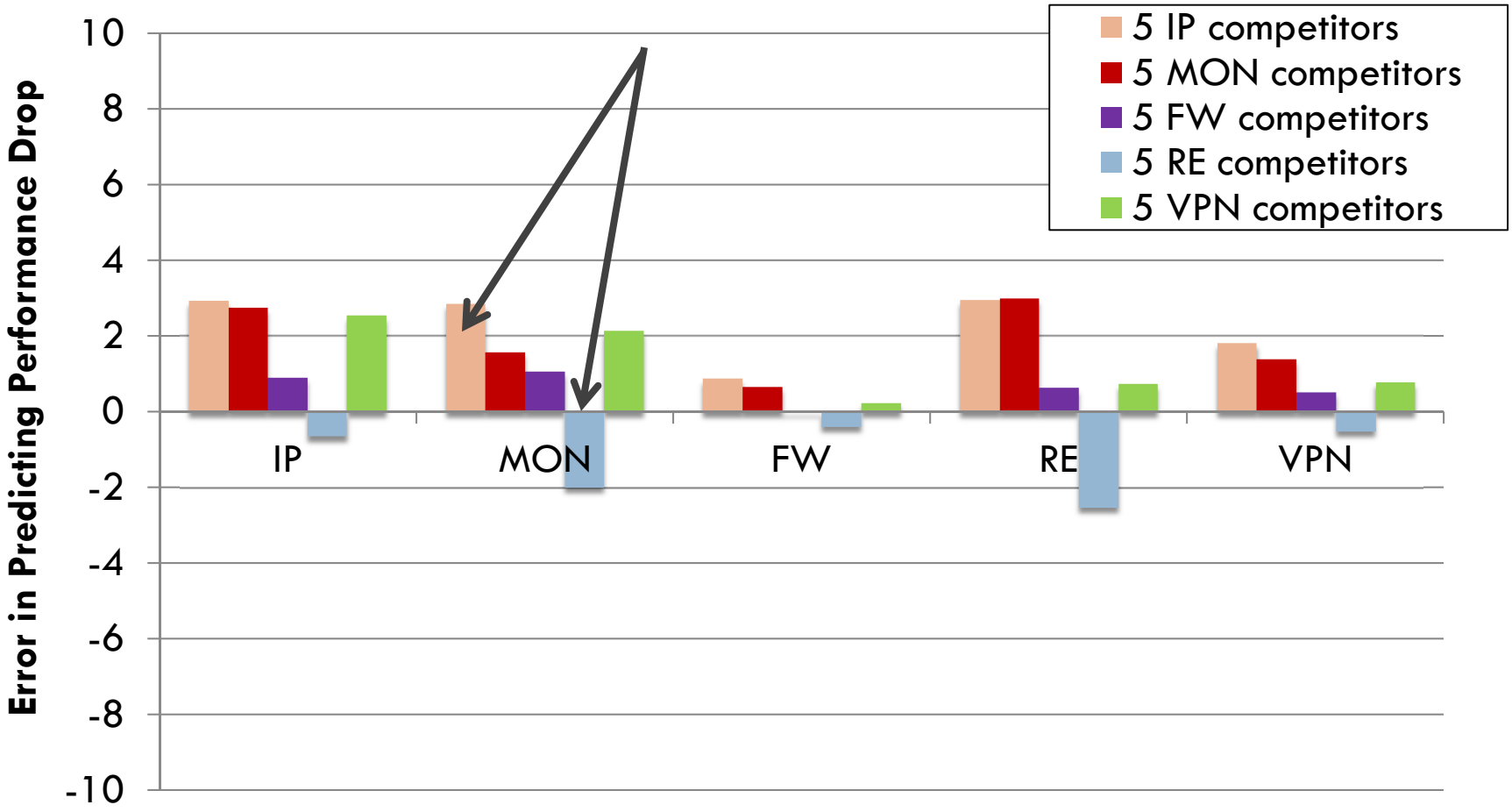


Simple offline profiling

Prediction Errors



Evaluation



Contention effects are predictable

Outline

- System overview
- Contention factors
- Observations on application behavior
- A simple prediction method
- Intuition

The Intuition

- Obs. #1: competitors' cache refs/sec determine drop
  - Aggregate data exceeds cache size
  - 3MB shared cache/core

The Intuition

- Obs. #1: competitors' cache refs/sec determine drop
  - Aggregate data exceeds cache size
  - 3MB shared cache/core
- Obs. #2: drop curve grows slowly after certain point
  - Most damage happens early on
  - Simple probabilistic analysis

Conclusion

- It is feasible to build a packet-processing platform with predictable performance using simple techniques
  - 3% prediction error
- Contention-aware job placement does not bring significant benefit to the overall performance
  - 2% potential improvement