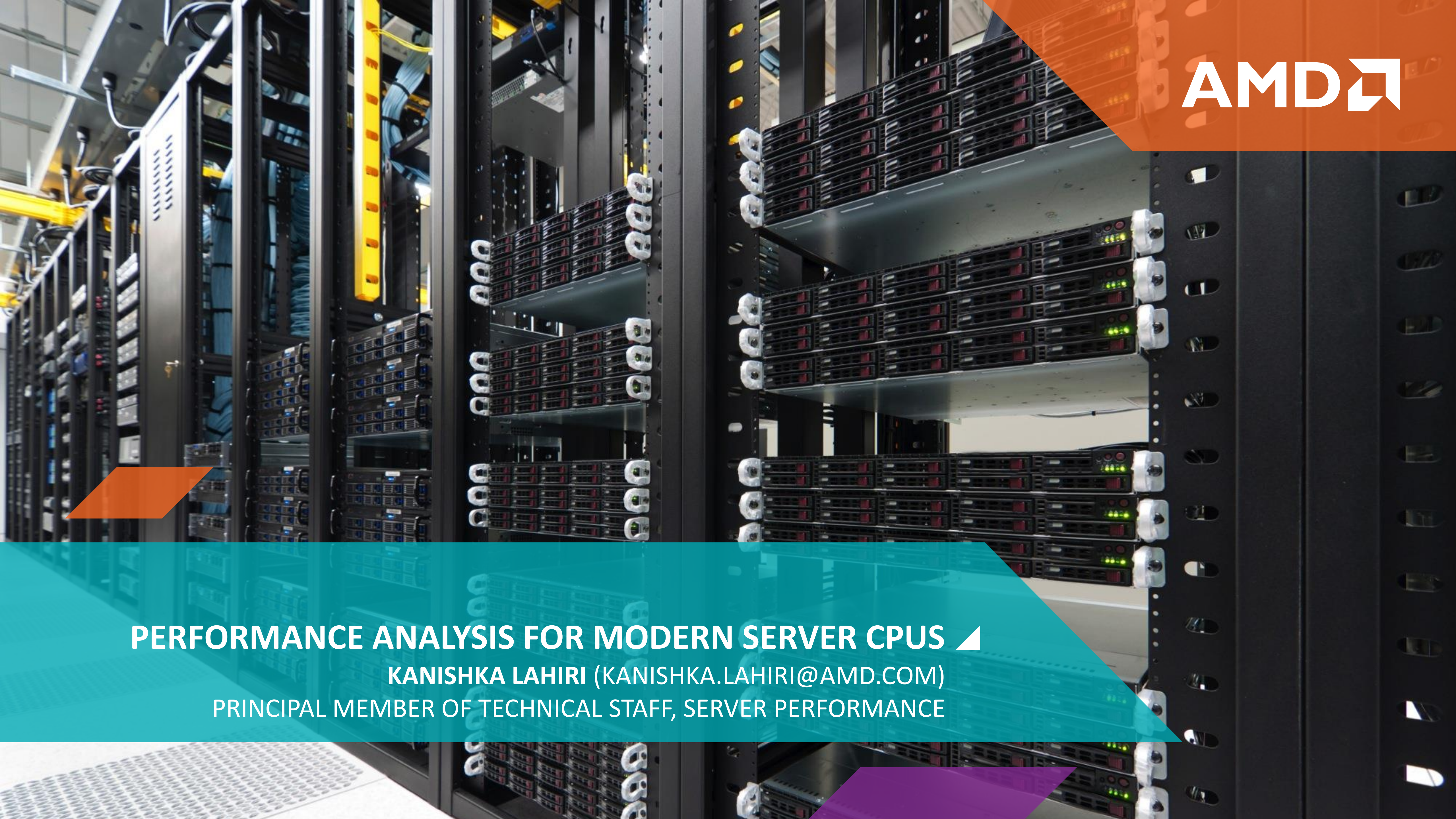


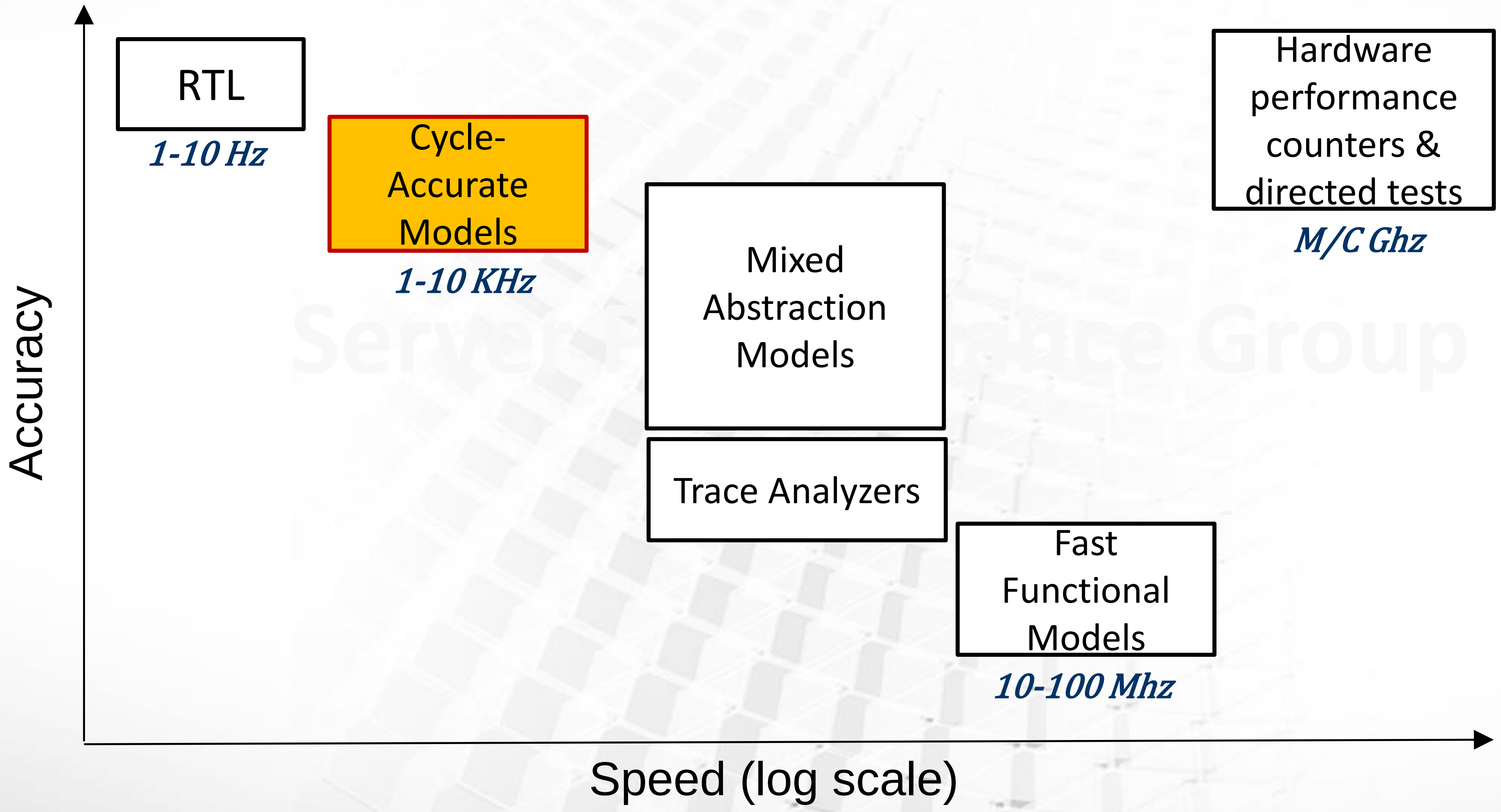
PERFORMANCE ANALYSIS FOR MODERN SERVER CPUS

KANISHKA LAHIRI (KANISHKA.LAHIRI@AMD.COM)
PRINCIPAL MEMBER OF TECHNICAL STAFF, SERVER PERFORMANCE

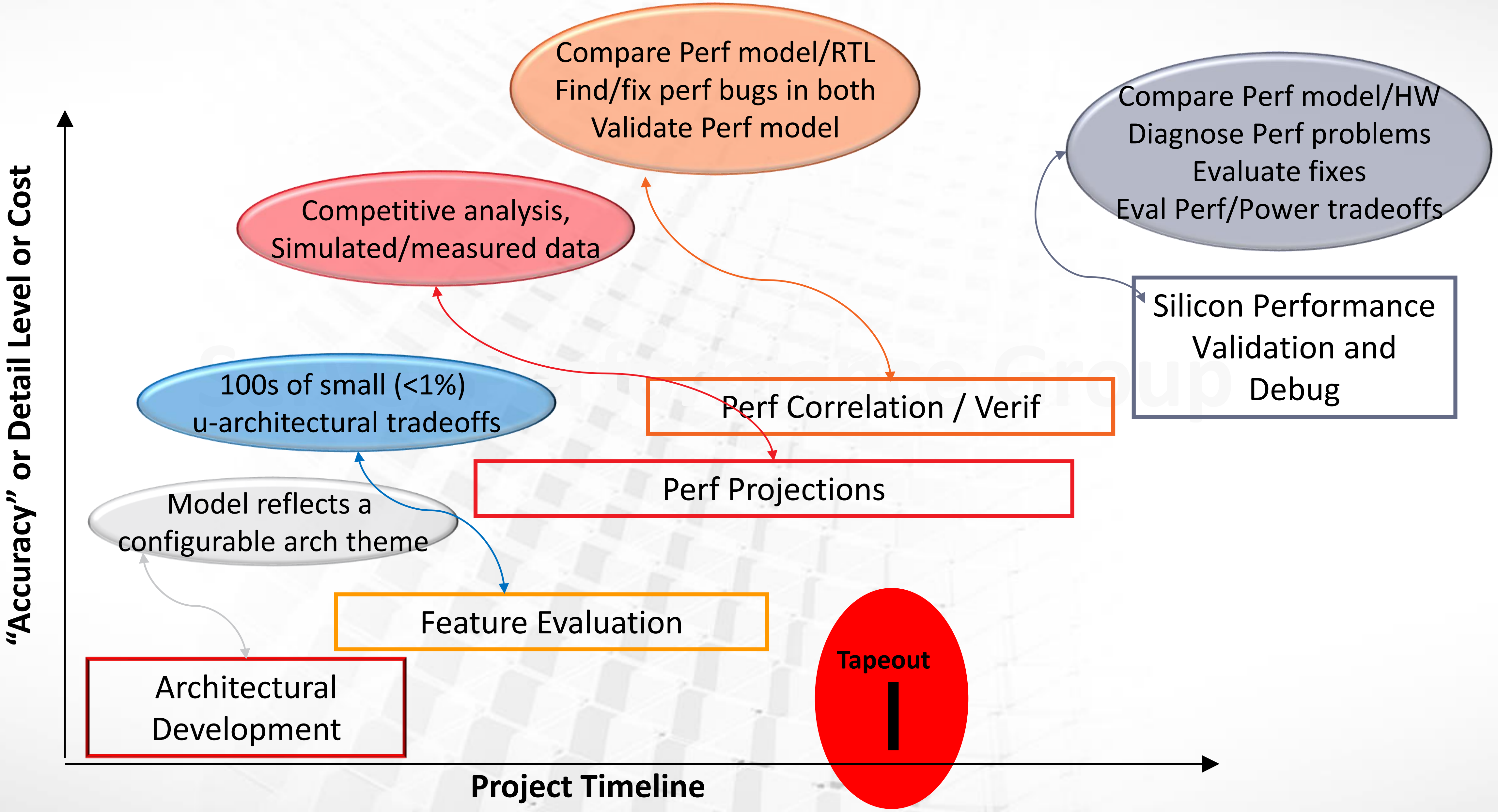
AGENDA

- Performance analysis and computer architecture
- Performance modeling methodology
- Server workload and CPU challenges

PERFORMANCE ANALYSIS TECHNIQUES



ROLE OF PERFORMANCE MODEL / ANALYSIS



CYCLE-ACCURATE PERFORMANCE MODELS

- C++ model with higher-level of abstraction than RTL
  - 100K lines of uarch specific code
  - 400K lines of shared infrastructure and library code
  - Highly parameterized at both the macro and micro level
  - Many, many configuration switches for structures, queues, algorithms, policies
  - Output: Hundreds of counters and statistics covering the uarch
  - Don’t model everything in the simulator (exceptions, power states, many rare conditions).

- Accuracy goal: match RTL (realistically 1-2%)
- Speed goal: as fast as possible (realistically ~10 Khz)
- Workhorse simulator for microarchitecture exploration/dev/correlation
  - Limited mT capabilities

SERVER WORKLOADS TO DRIVE PERFORMANCE MODELING

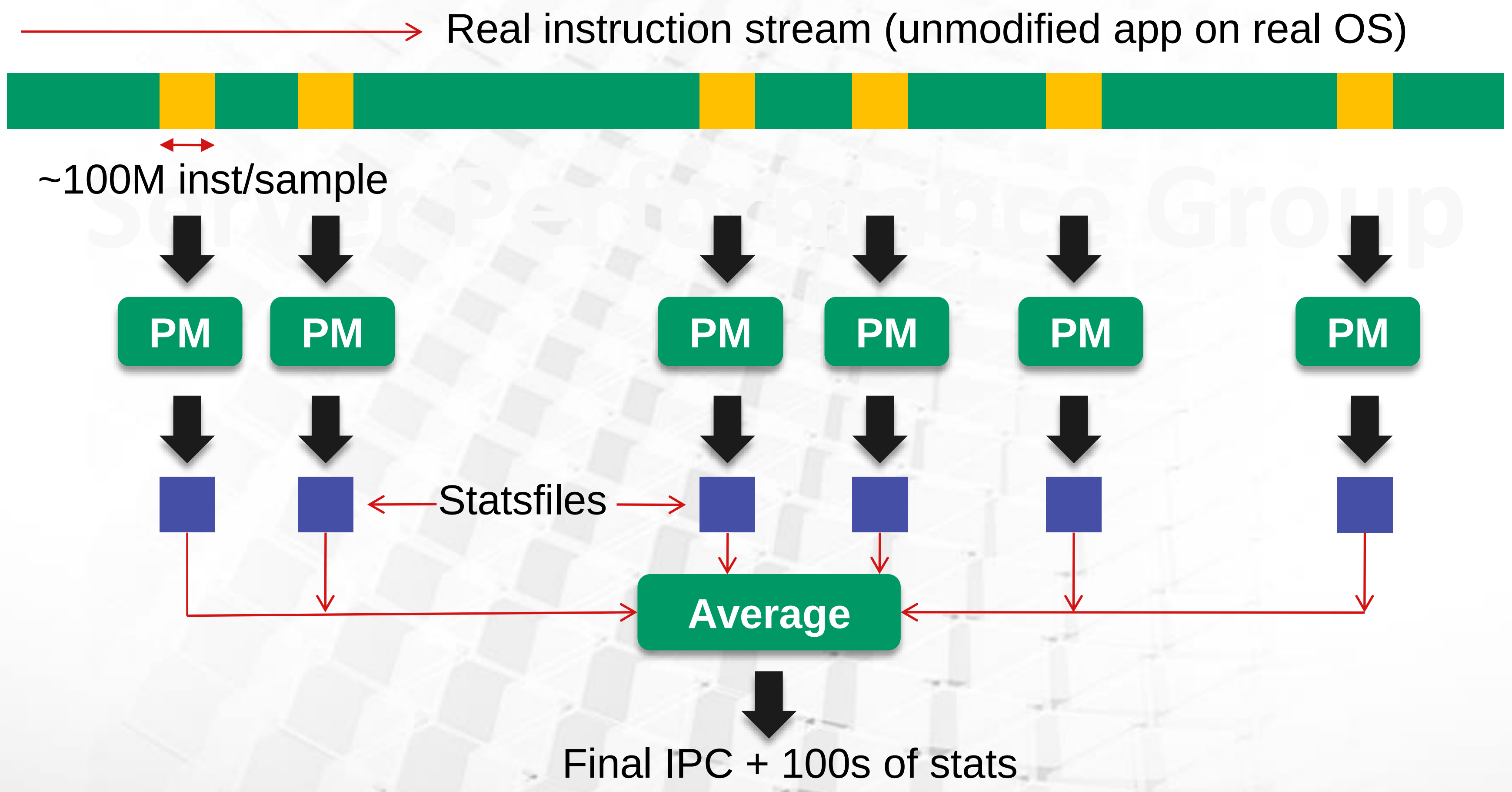
- SPEC CPU2006/2017
  - Workhorse CPU throughput/speed benchmark
  - Gcc / optimizing compilers
- Enterprise (Classic)
  - SPECJbb15 (Java)
  - Traditional Data bases (TPC-C/TPC-E)
- Cloud
  - Spark / Hadoop
  - NoSQL databases
  - Machine learning

*A great model is useless without proper workloads to drive it. Representativeness is **key***

- High-performance computing
  - DGEMM (matrix multiply aka HPL), FFT
  - LS-Dyna3D, Ansys, …
- Virtualization
  - SPECVirtSC 2013
  - VMmark (from VMware)
- Microbenchmarks
  - Latency, bandwidth

REAL WORKLOAD SIMULATION & SAMPLING

- Cycle accurate simulators run at ~10 Khz
- Simulating SPEC CPU 2006 in entirety (44T dynamic insts) would take 100+ yrs


Real instruction stream (unmodified app on real OS)
~100M inst/sample
PM
PM
PM
PM
PM
PM
Statsfiles
Average
Final IPC + 100s of stats

TRACING TOOLS

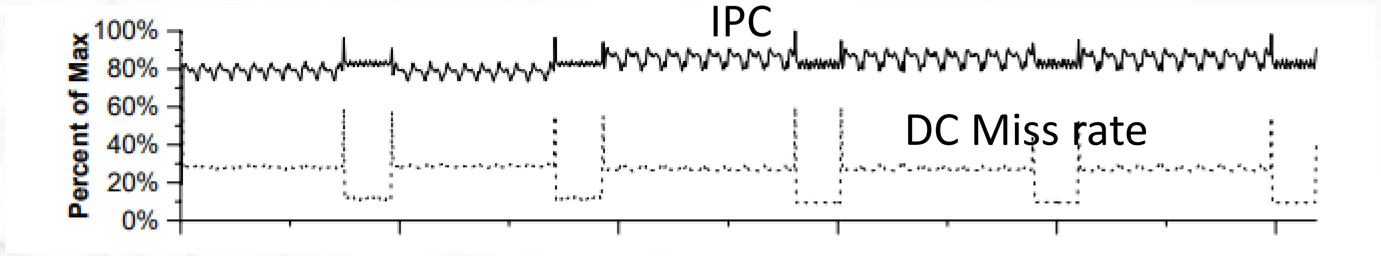
- Leverage fast functional simulators
  - Virtual platform of a PC/server
  - Boot unmodified OS, run applications
  - Examples: QEMU, AMD SimNow

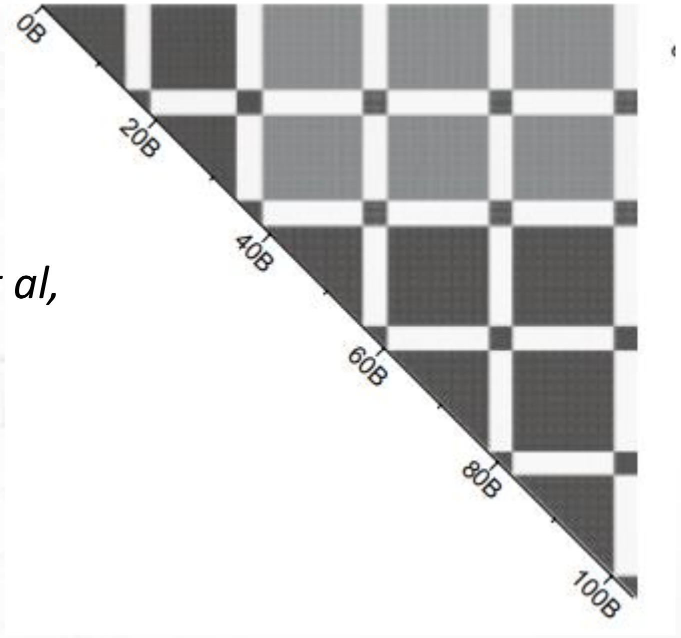
- Hardware based tracing tools
  - Needs proprietary custom hardware

- Dynamic binary instrumentation tools
  - Popular for program analysis
  - Injects profiling code into running binary
  - User space only
  - Examples: DynamicRIO, Valgrind

CHOOSING SAMPLE POINTS

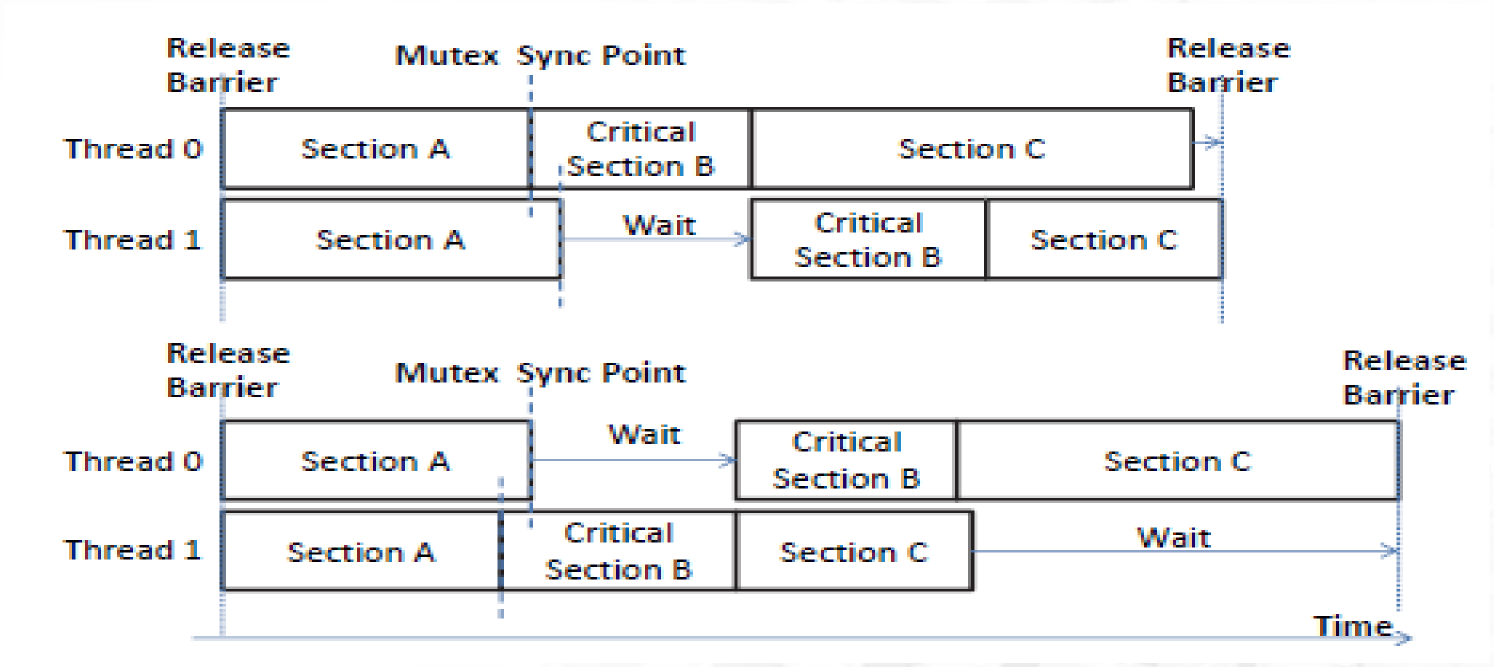
- Exploit phase behavior in programs
  - Profile program stats over fixed intervals (e.g. BBVs)
  - Run clustering
  - Choose one trace / input per cluster
- **<u>Dramatically</u>** reduces the no. of inst simulated (<0.5% for SPECINT)
- Widely used in academia and industry (with variants)
- Sample points:
  - Fixed lengthtraces
  - State snapshots





Source: Sherwood et al, ASPLOS02

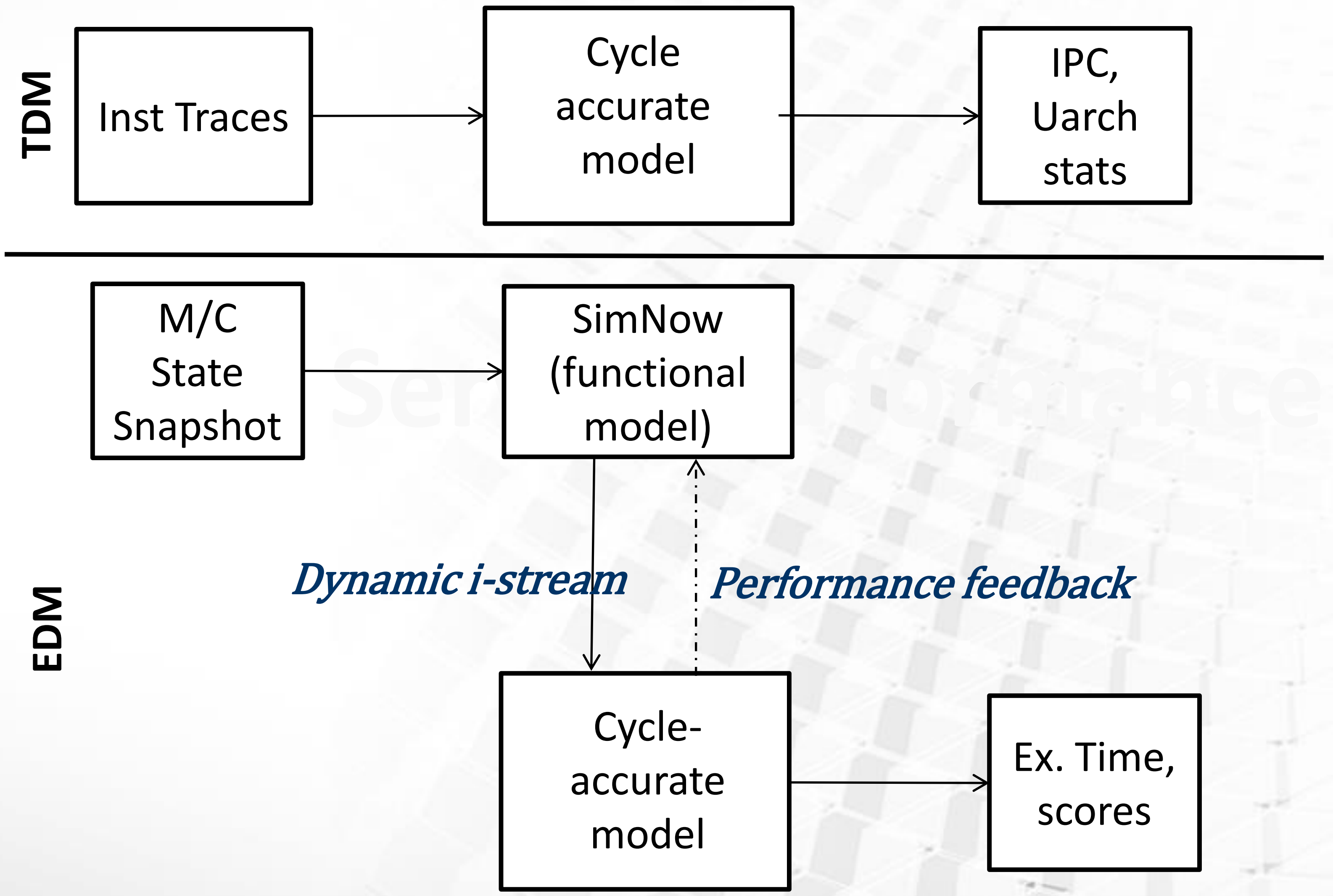
NON-DETERMINISM IN SERVER WORKLOADS



Source: SynchroTrace: ISPASS 2015

- Need environment where SW adapts to performance changes
- IPC doesn’t make sense any more. Need to estimate Wall Clock Time

TRACE DRIVEN V EXECUTION DRIVEN MODELS



TDM VS EDM TRADE OFFS

Trace Driven

Pros

- *Faster*
- ***Deterministic i-stream***
- *Easier to sample*
- *Easier to validate*

Cons:

- *No wrong path instructions*
- ***Deterministic i-stream*** *(doesn't model mT effects)*

Execution Driven

Pros:

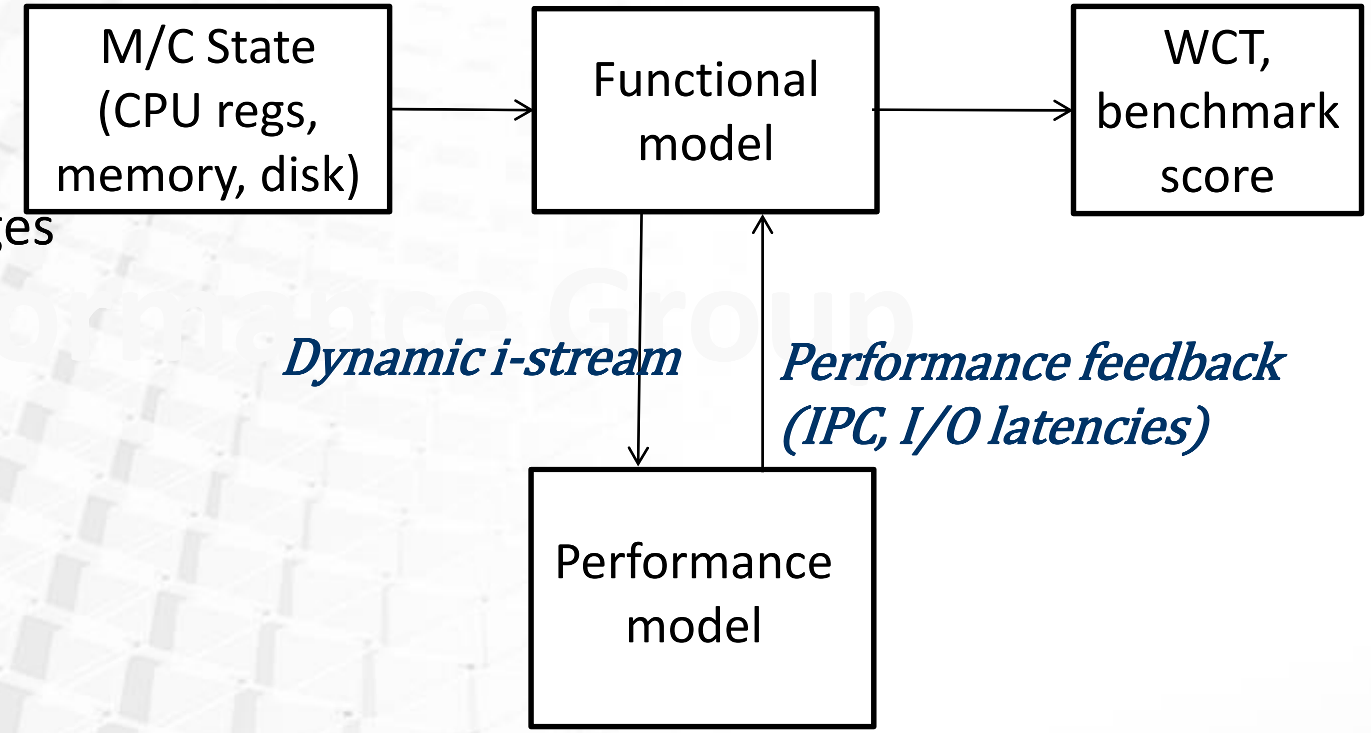
- *Wrong path*
- ***More realistic i-stream***

Cons:

- *IPC is not a good metric, need WCT*
- *Sampling less understood*
- *Complexity*
- *Speed*
- *Validation*

POTENTIAL SOLUTION – 1 : EXECUTION DRIVEN W/PERF FEEDBACK W/SAMPLING

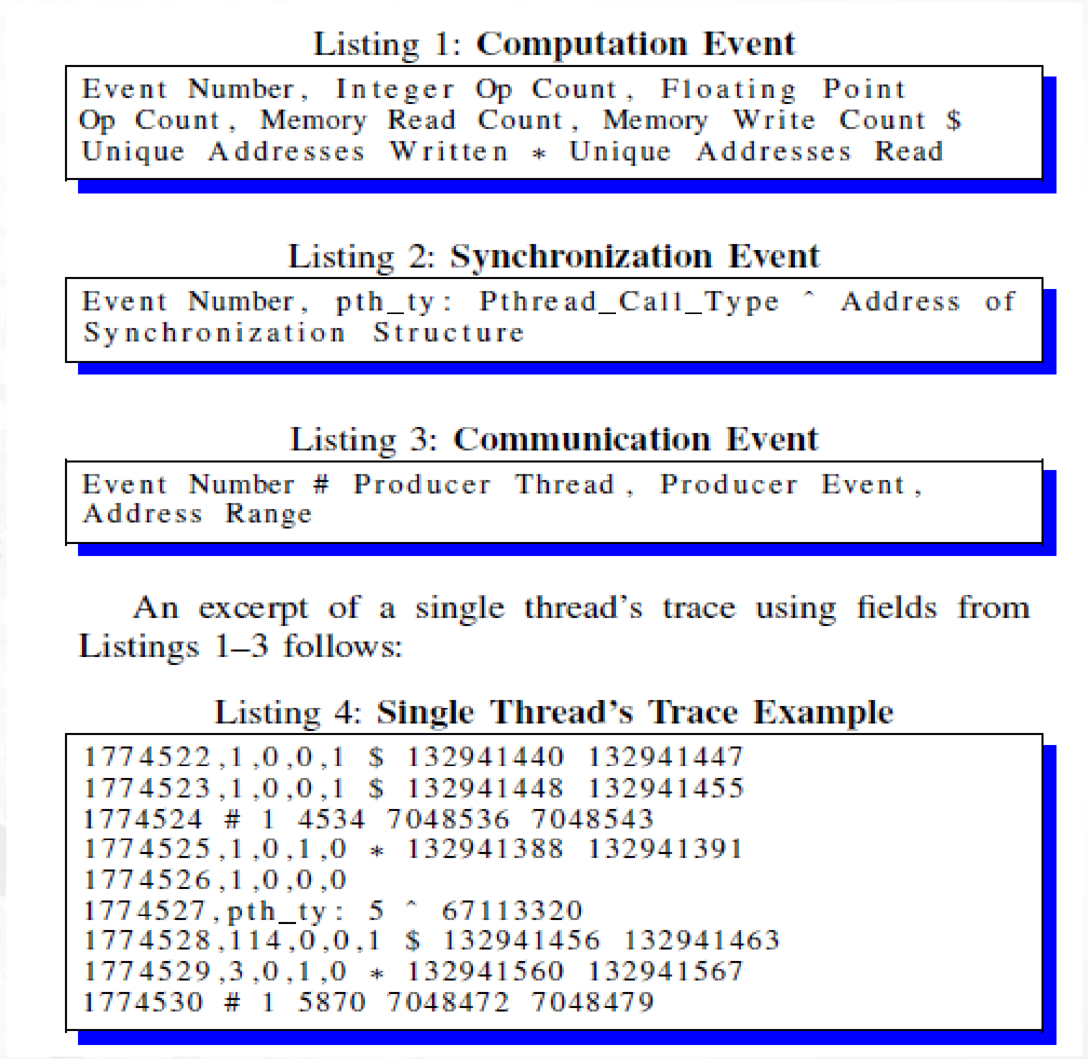
- Dynamic sampling techniques can help (Falcon et al, ISPASS 2007)
- **Step 1:**
  - Run functional simulation, use low overhead profiling to detect phase changes (E.g. BBV profiling, Online clustering)
- **Step 2:**
  - Reuse perf from previously seen phases
- **Step 3:**
  - Take a sample on a new phase (engage timing model, run EDM simulation)
- Drawback: serialized simulation
  - Run time, Health


M/C State (CPU regs, memory, disk)
Functional model
WCT, benchmark score
Dynamic i-stream
Performance feedback (IPC, I/O latencies)
Performance model

POTENTIAL SOLUTION – 2: SYNCHRO TRACE (ISPASS 2015)

- Encodes dependency and synchronization information into the trace
- Sacrifices details (exact instructions)

- Replies on trapping on calls to specific sync primitive (e.g. pthreads)
- Limited to user space synchronization (limitation of tracing infra)



Listing 1: **Computation Event**

```
Event Number, Integer Op Count, Floating Point
Op Count, Memory Read Count, Memory Write Count $
Unique Addresses Written * Unique Addresses Read
```

Listing 2: **Synchronization Event**

```
Event Number, pth_ty: Pthread_Call_Type ^ Address of
Synchronization Structure
```

Listing 3: **Communication Event**

```
Event Number # Producer Thread, Producer Event,
Address Range
```

An excerpt of a single thread's trace using fields from Listings 1–3 follows:

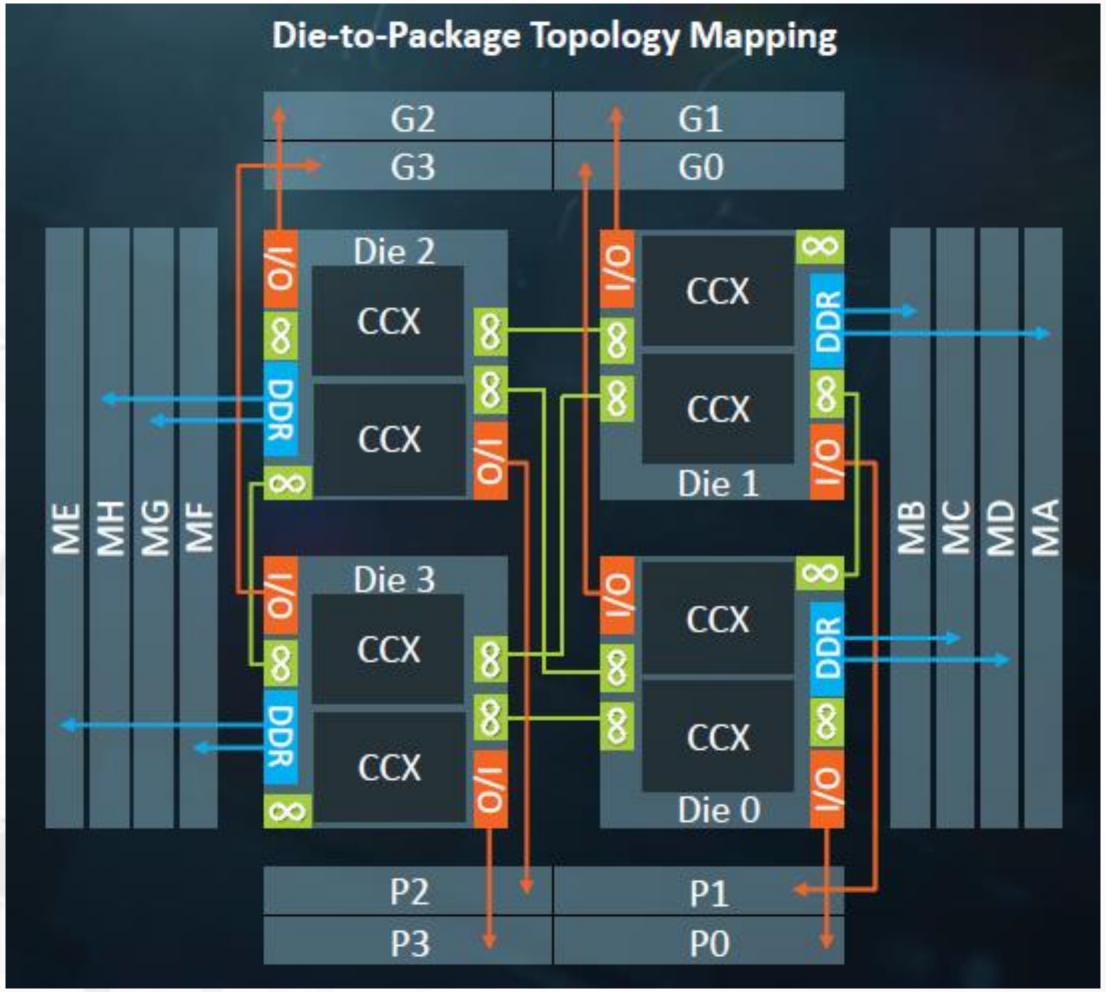
Listing 4: **Single Thread's Trace Example**

```
1774522,1,0,0,1 $ 132941440 132941447
1774523,1,0,0,1 $ 132941448 132941455
1774524 # 1 4534 7048536 7048543
1774525,1,0,1,0 * 132941388 132941391
1774526,1,0,0,0
1774527,pth_ty: 5 ^ 67113320
1774528,114,0,0,1 $ 132941456 132941463
1774529,3,0,1,0 * 132941560 132941567
1774530 # 1 5870 7048472 7048479
```

MODERN SERVER CPU EXAMPLE

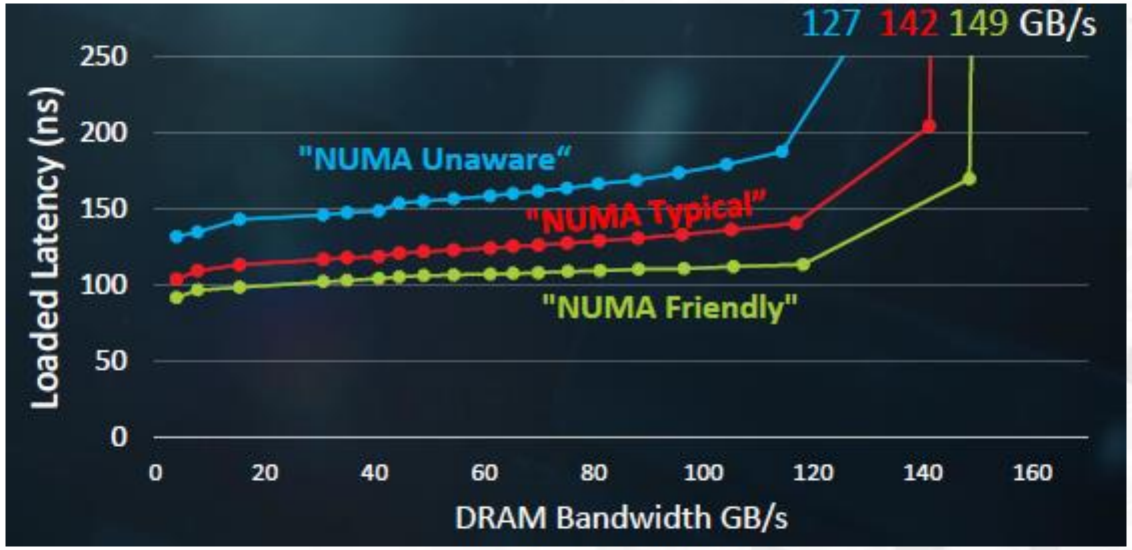
AMD EPYC^{TM} 7601

- Each CCX is a 4C/8T complex
  - Each core has private L2
  - Each CCX has a shared 8MB L3
- Each die is a 8C/16T SCM
  - 16MB of L3
  - 2 DDR channels
- Each socket is 4 dies:
  - 32C/64T
  - 64MB L3
  - 8 DDR channels
- Cycle accurate simulation simply does not scale

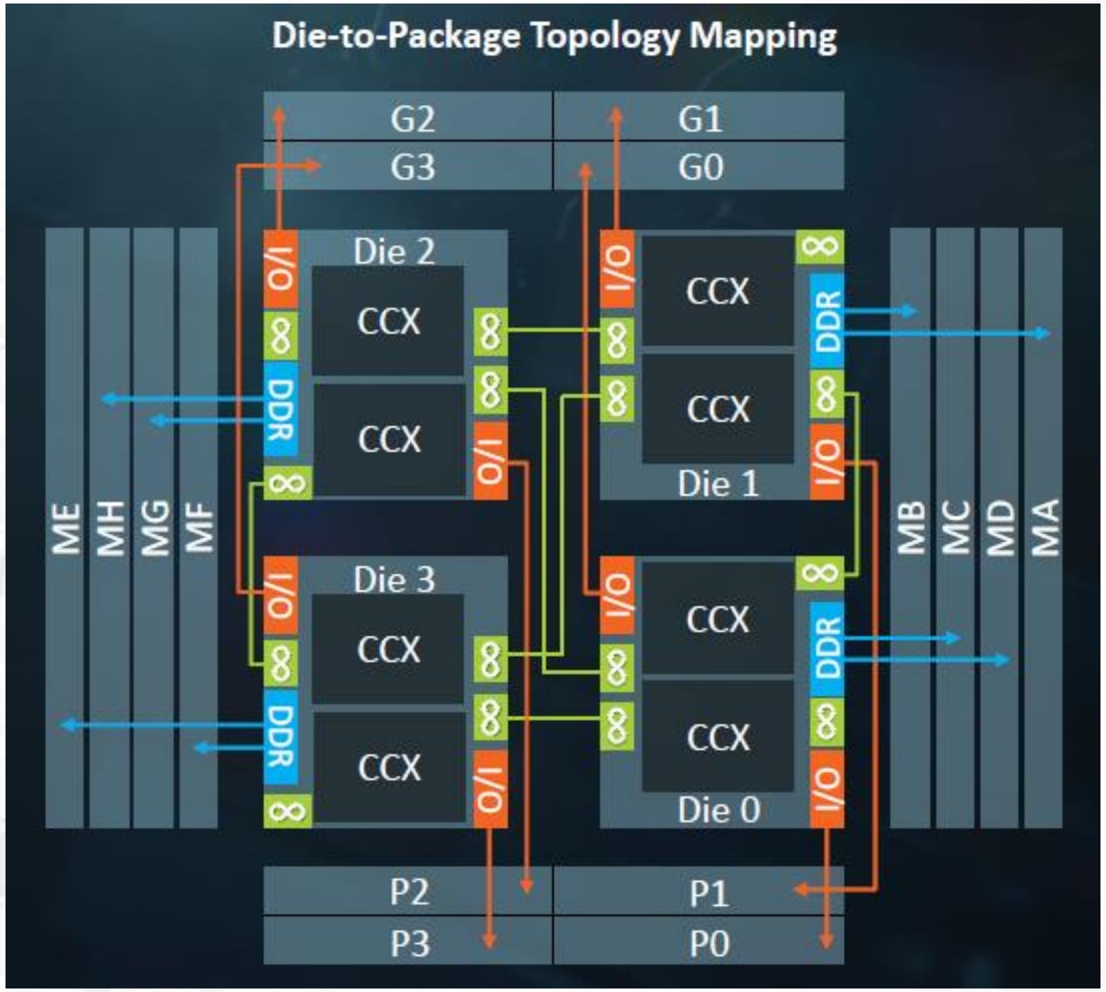


SIMULATING COMPLETE SERVER SOCS

- Need to model loaded latency

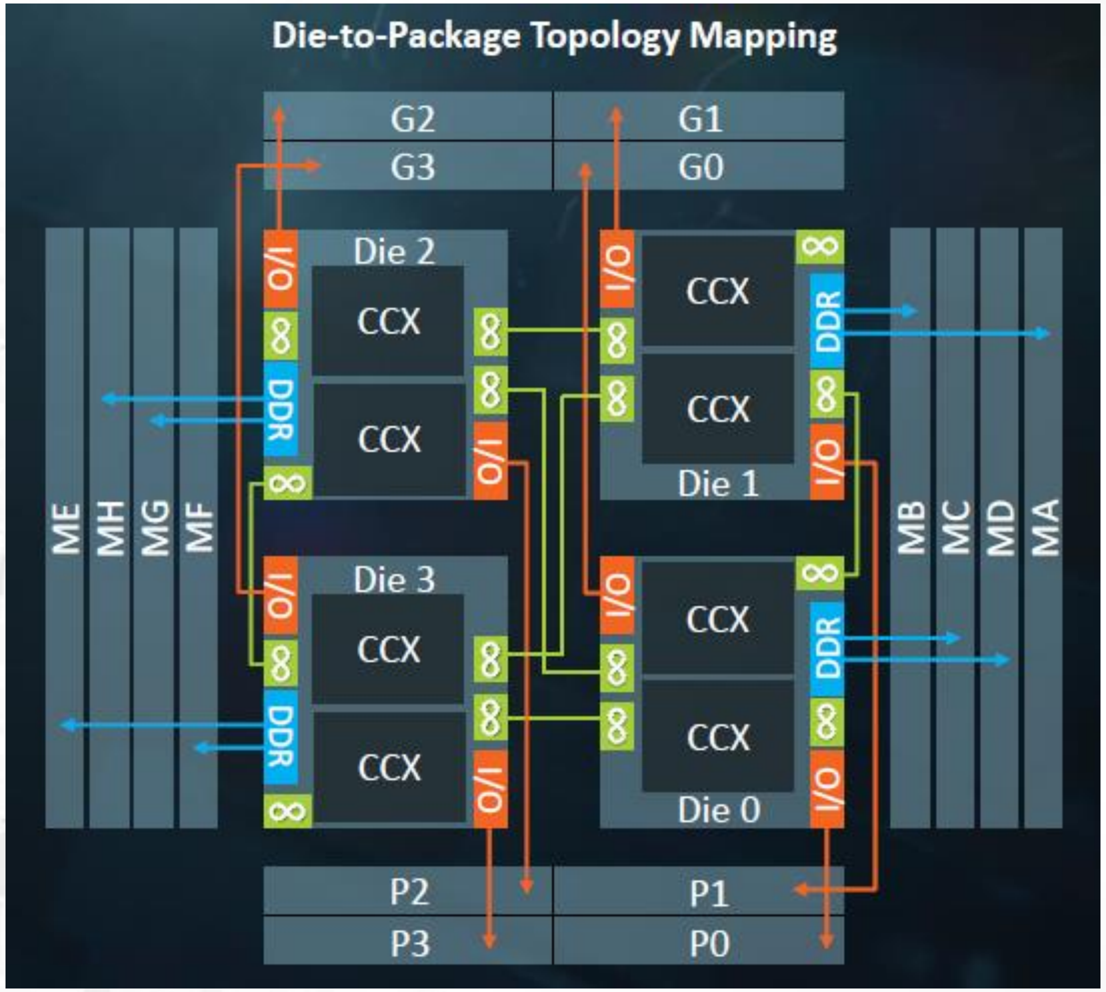


- Running 64 threads in the perf model is not practical



SIMULATING COMPLETE SERVER SOCS

- Use lower thread counts
- Mix abstraction models
- Bus trace driven models



SIMULATING LARGE CACHES: SOLUTIONS

- Why not simply warm up caches for longer?
  - Instruction trace lengths are limited
  - Storage is a challenge
  - The problem of legacy traces

- Research directions:
  - Check point cache state (very accurate, but uarch dependent) [Lauterbach et al, SUN, 1993]
  - Stitch (take previous sample's state) [Kessler, IEEE Trans Computers, Iss 43]
  - Combine stitch with fraction of new sample [Conte 1996]
  - Use reuse distance distros to estimate hit/miss rates [WarmSim/CacheSim, ISPASS 2014,16]
  - Use an estimated miss rate to statistically warm up the cache

SUMMARY & FUTURE AREAS OF RESEARCH

- Performance analysis is critical
  - Variety of tools
  - C++ based modeling is key
  - 10% inspiration, 90% perspiration
- Cycle accurate CPU models are critical
  - Model all the details and more of a uarch
  - Stimulate with proper workloads
  - Aggressively downsample
- Server CPU challenges:
  - Modeling non-determinism
  - High thread counts
  - Warming up large caches

- Near term challenges for performance modeling for server
  - Ever more aggressive sampling
  - Warm up techniques
  - Execution driven challenges
  - Synchronized traces (ISPASS 2016)

- Longer term challenges
  - Model/predict cluster level performance
  - Low utilization workloads
  - Exploit multi core architectures

DISCLAIMER & ATTRIBUTION