Benchmarking the Ethernet-Federated Datacenter

M. Gusat, C. DeCusatis,
C. Minkenberg, L. McKenna,
K. Bhardwaj

G. J. Paljak, A. Pataricza,
I. Kocsis

IBM Research, Zurich Research Laboratory
IBM Systems and Technology Group, Poughkeepsie
IBM Tivoli, Australia Development Laboratory

Fault Tolerant Systems Research Group
Budapest University of Technology and Economics

Overview

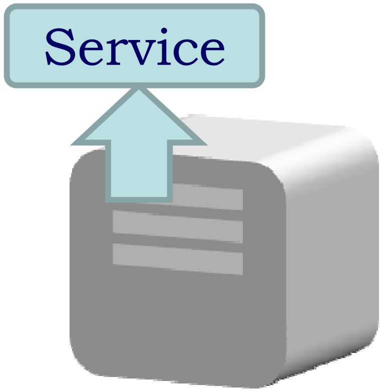
- Why DC Benchmarks?
  - networking in scalable DC and Clouds

- The 3 pillars of DC benchmarking
  1. Workload characterization and monitoring
     - workload, traffic, metrics
     - transaction tracing
  2. Simulation
     - Mercury/Venus environment
     - QCN case study
  3. Analytics
     - DC formulated as a feedback control problem

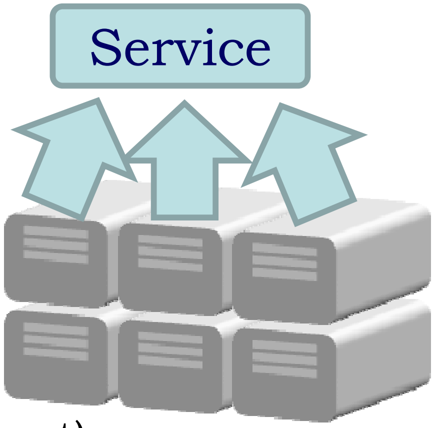
- Conclusion

Note: Animated features not available in the PDF version.

Benchmarking (Cloud) Datacenters



- **Traditional DC**
  - **low cost** of computing power
  - mostly **under-utilized** servers, storage, and network



- **Cloud Computing DC**
  - apps provided as standardized offerings
  - run on **virtualized resources**
  - **dynamically reconfigurable** resources

- **DC Network is the key**
  - LAN (Ethernet), StAN (FibreChannel), SAN (InfiniBand, Myrinet) converge into one integrated Datacenter Network fabric of 10-100 Gbit/s

- **Aim: Holistic view of the Cloud DC, with rigorous benchmarking and monitoring.**

Why DC/Cloud Benchmarks?

- Can’t improve what we can’t measure

- Today’s DC performance is not measurable beyond W and %
  - TPC-xyzW benchmarks are box-oriented, neither DC nor Cloud
  - HPC benchmarks are system-oriented, but not valid for commercial workload

- DC/Cloud operators care about power, cooling and utilization
  - average server utilization: 12-17% [IDC, Gartner]
    - Amazon wants > 80%
  - DC network utilization?

- VM: seems the key to better (server) utilization, yet…
  - Google doesn’t use virtualization
  - How about the network: Is VLAN the equivalent of server VM?
  - Does VM serve the customer or the operator ?

- Customers care about SLA … [+ privacy, phy. separation, price, mgnt.]
  - SLA: measured service ➔ managed demand
    - e.g. algorithmic/arbitrage trading measure latency in M$, not us…

Characterizing the Commercial/Analytic DC Workload

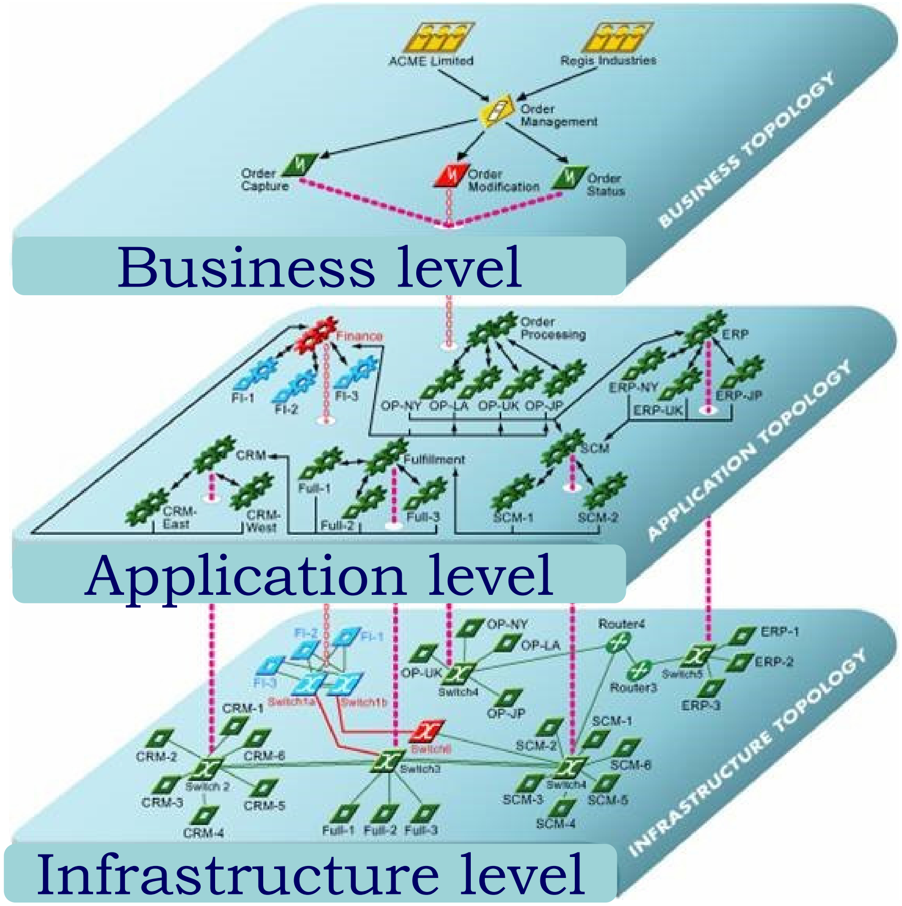
- **Mapping business transactions to application and infrastructure level events** (to functions calls, network packet transfers)
  - Assemble **causal paths,**
  - Observe **distributions**,
  - Detect **patterns**

- Essential to understand the system and to observe how the changes in workload affect internal system functionality.

- Challenge

  I. **No de-facto standard** as MPI in HPC, many protocols in a commercial DC: HTTP, CORBA, JDBC, etc

  II. Correct, instance-level **causal path reconstruction** in a typical multi-tier, heterogeneous architecture **is difficult**



Monitoring and workload characterization

- Commercial DC workloads are not quantitatively known (yet)

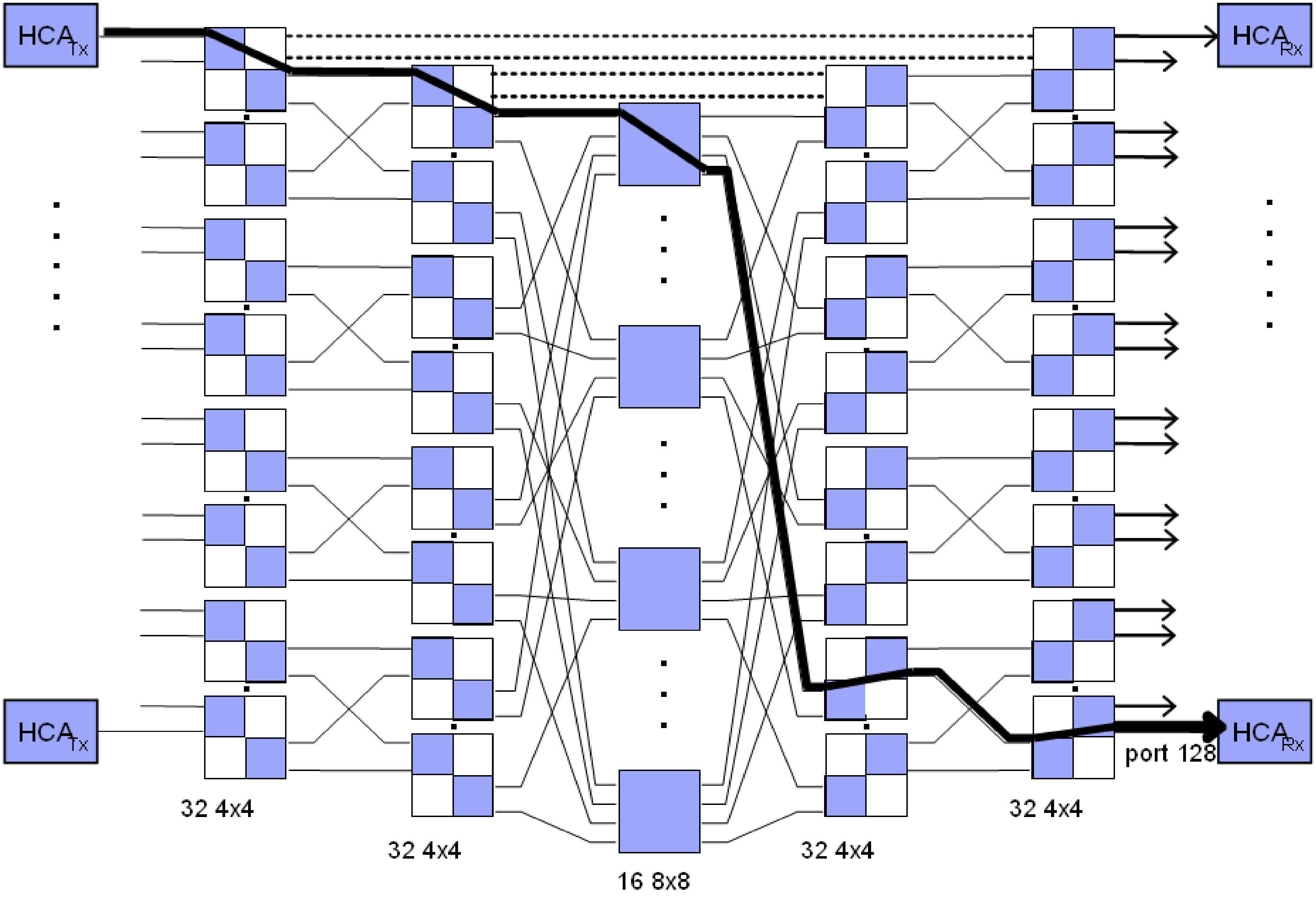
- Wanted
  - Agreed [observable] **workloads**
  - Instrumentation and **monitoring** tools
  - Detailed performance **models** for the **<u>full DC</u>**

- Acceptable mid-target: Realistic traffic generator…
  - Wherefrom?
    - traces or exec-based models

- Obstacle: DC traffic space-time distribution = ?
  - requires estimation of TopN causal paths (essential transactions)

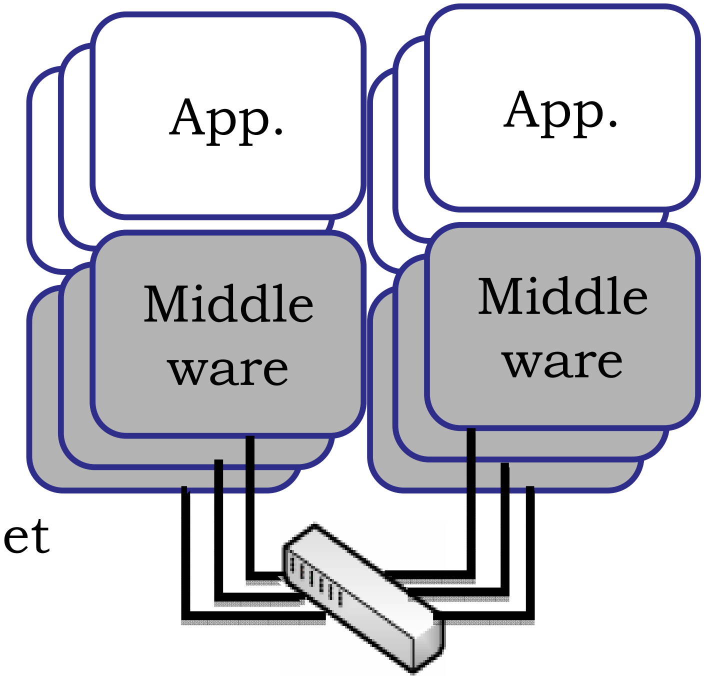
Tracing a Day in the Life of a DC Transaction

Projecting a Transaction on the DC Network (DCN)



Review: White box approaches

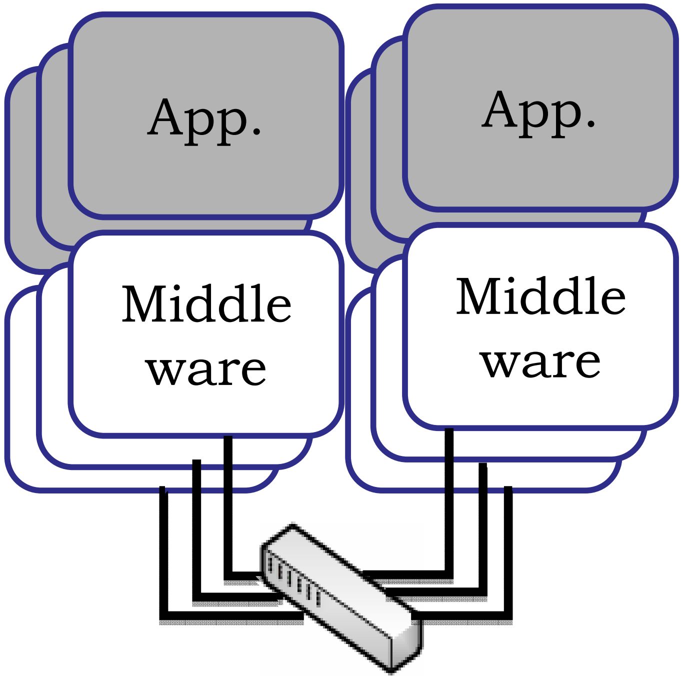
- **Application source code can be modified**
- Tagging transactions with IDs
  - register tag and timestamp when transaction passes boundaries
- Examples:
  - NetLogger (Gunter at al., Berkeley Lab)
  - WebMon: web transaction monitoring with custom JavaScript and cookies for id storage (Gschwind et al, HP Labs)
  - User Programmable Virtual Networks (Meijer et al.)
  - ARM: open standard defining interfaces to insert and track identifiers (http://www.opengroup.org/management/arm)



Review cont': Gray box approaches

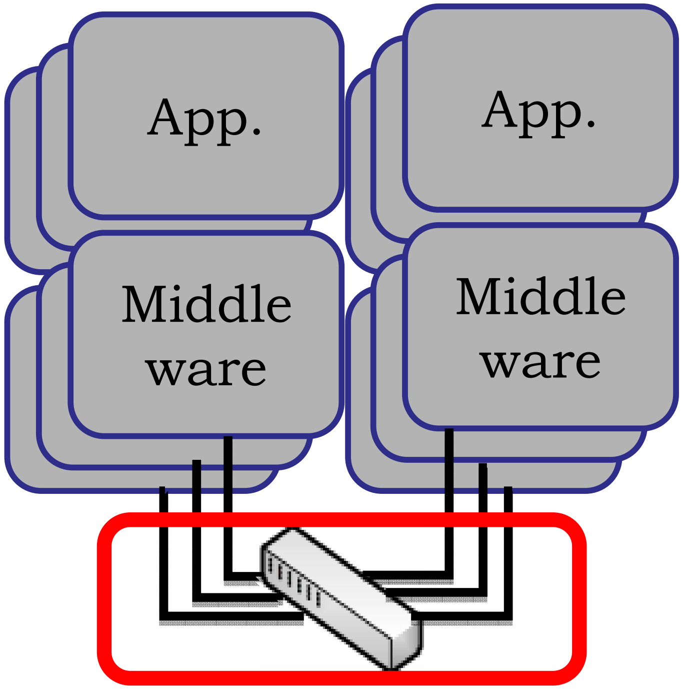
- The **middleware platform** keeps track of transactions by **built-in interceptors**, JIT-compiled tracing instructions

- Examples:
  - Magpie (Microsoft Research), built on the Event Tracing for Windows framework with additional platform modifications
  - IBM's ITCAM for Transactions with ARM-based transaction tracking
  - Java 2 EE component instrumentation (PinPoint, Chen et al.)
  - Java VM agent-insertion (Mirgorodskiy et al.)
  - Various approaches using CORBA interceptors (Moe et al.; Debusmann et al.; Li et al.)
  - Near system kernel solutions: observe processes communication through shared memory, „data tagging" (Whodunit, Chanda et al.; Mysore et al.)

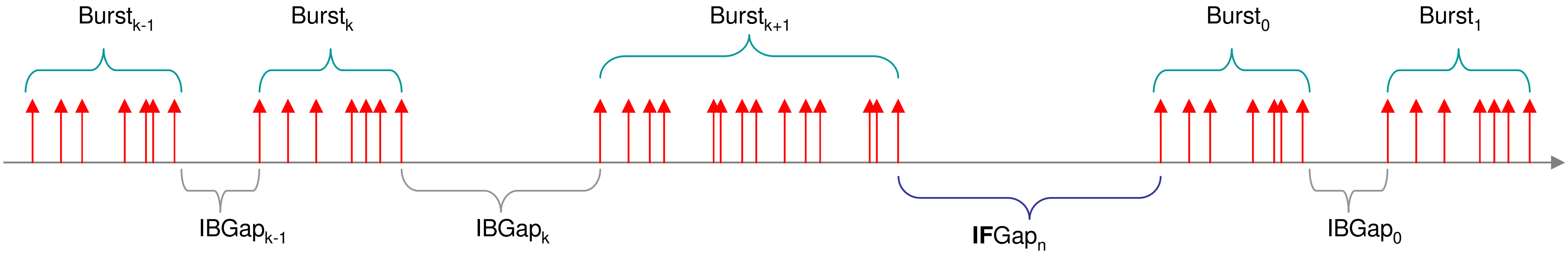


Review cont': Black box approaches

- Only **network traffic** is observed (passively)
  - Probabilistic models
- Examples:
  - Aguilera et al. (HP Labs)
    - **1st method:** assume: call-return message pairs; find possible parent-child relations and evaluate likelihood
    - **2nd method:** consider message traces as time signals; apply convolution to find messages with similar time shift
  - Anandkumar et al. (IBM Watson)
  - assume independent, identically distributed servicing times,
    - and use maximum-likelihood to match events and re-build transactions



Metrics for Workload Evaluation



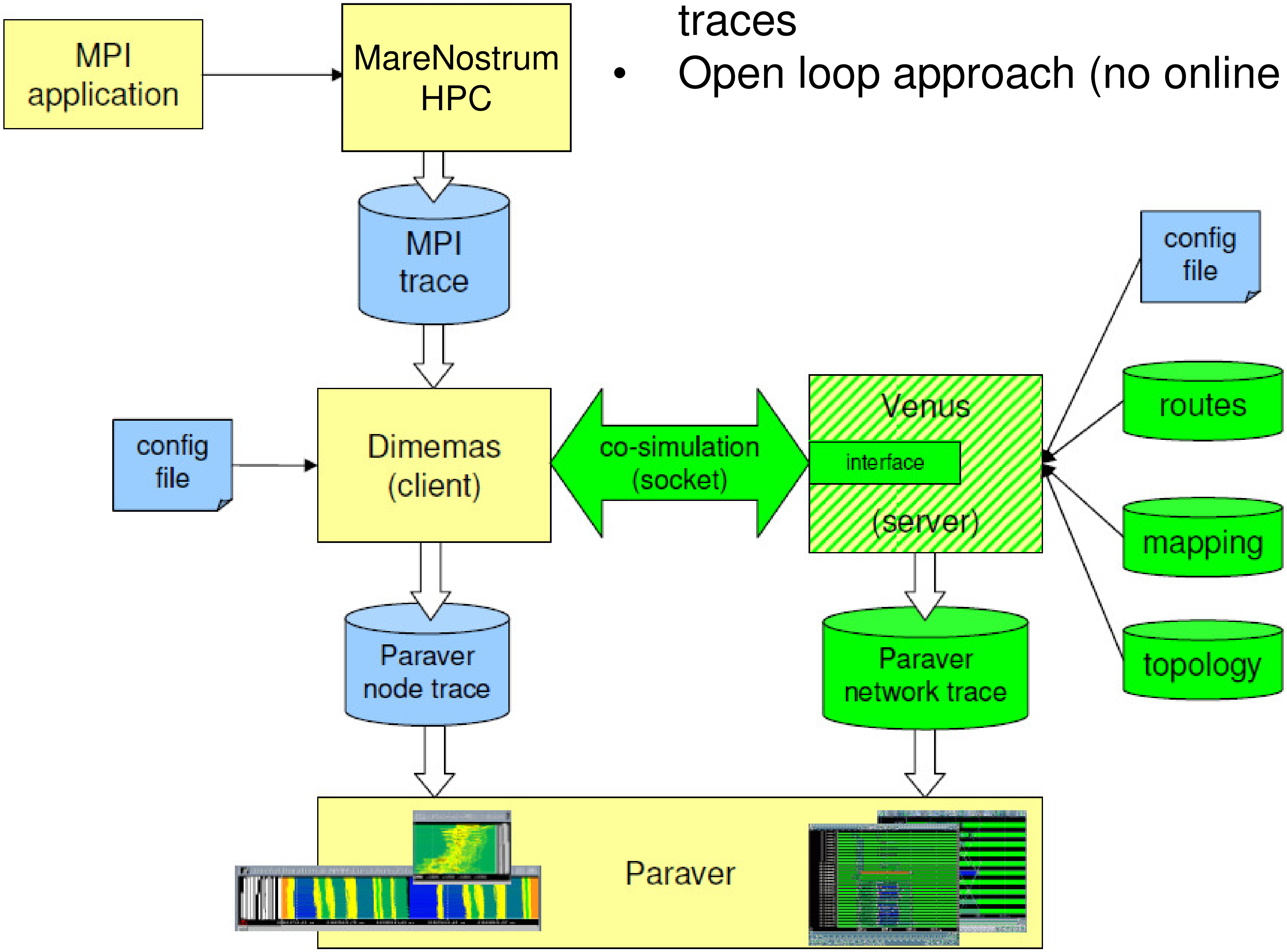
- Workload is a set of the following components, listed hierarchically: **work, job/transaction, flow, burst, and packet (frame, flit).**
- Established **DCN level metrics**:
  - latency (end-to-end (e2e) delay at L7), throughput, jitter, fairness
- Is **Flow Completion Time (FCT)** a good metric?
  - Yes, but sensitive to distributions
    - For example, for Pareto distributions, FCT loses its relevancy because $FCT = \Sigma(t_{\text{inject}} + t_{\text{queue}} + t_{\text{flight}} + t_{\text{RTX}}) \neq L_{\text{e2e}}(X)$, i.e., the central limit theorem does not apply
  - L2 detail of FCT: Flow definition?
    - (1) flows received entirely without any loss;
    - (2) flows received entirely with some loss;
    - (3) flows received partially,
    - (4) flows not yet having arrived at destination.

Simulation for Large-scale Systems

- „Monte Carlo” discrete even simulations
  - synthetic workloads, based on predetermined probabilistic distributions drive accurate functional models
- Trace driven simulations
  - computing nodes are represented by a trace of two kinds of records:
    - computation: wait time
    - communication: fed into the DCN model
- Used in HPC
    - Message Passing Interface (MPI) is a de-facto communication standard
    - MPI libraries instrumented for tracing are available
    - IBM: MARS, VENUS etc. simulation environments

2nd Generation HPC Simulator: Venus-Dimemas

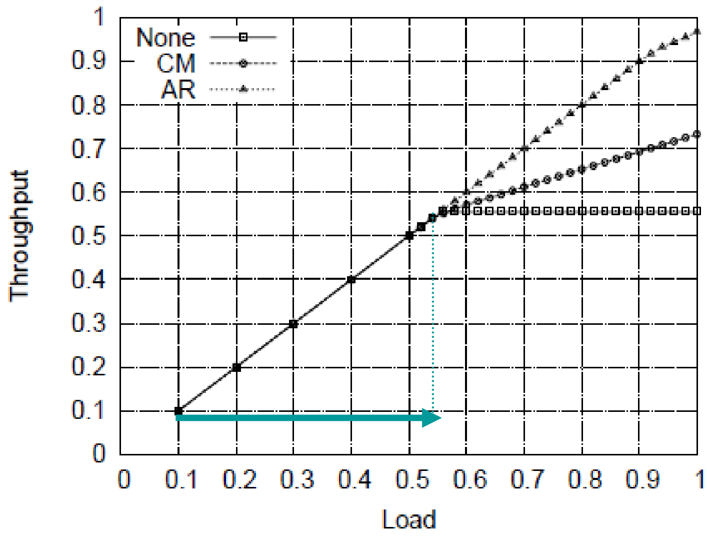
- System models for simulation based on message traces
- Open loop approach (no online feedback)

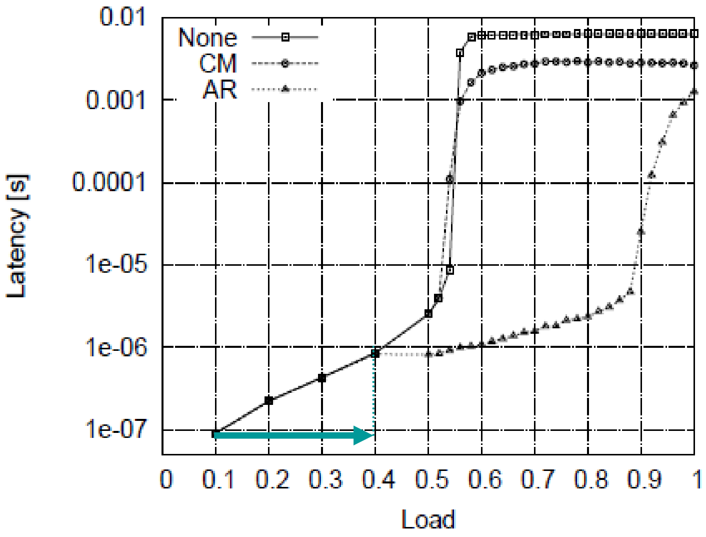


DCB Case Study: 802.1Qau-based Adaptive Routing

CEE Question: Is QCN Needed?

- DCB/CEE: Ethernet + PFC + ETS + QCN + upcoming
- Detroit: Prios->V8; PFC->hand-brake; ETS -> gearbox; QCN -> ABS
  - …where's the Steering/Suspension ?
- Change of DCB Congestion Management: use the QCN load sensor
  - (i) to steer: balance the load by adaptively routing (AR) the traffic around the transient hotspots (absorb shocks);
  - (ii) *iff* all admissible multi-pathing options have been exhausted *and* if the hotspot **persists**, then enable source-injection rate control (ABS)



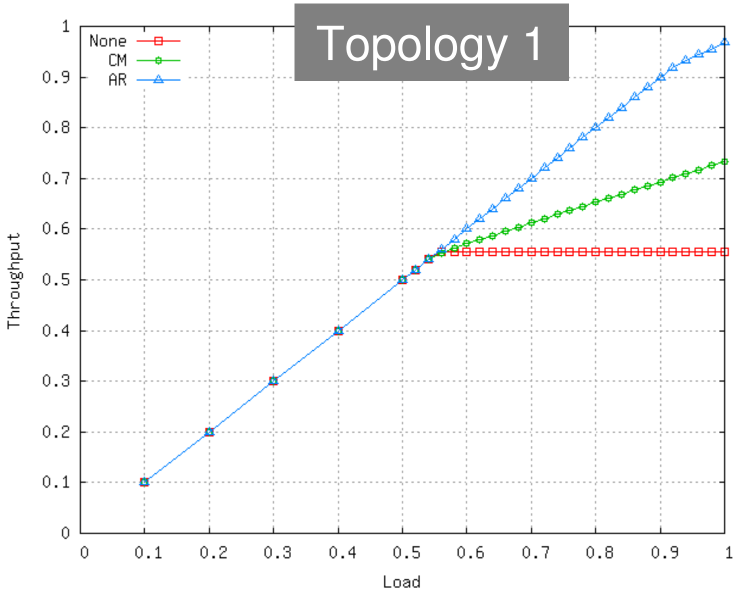


A: QCN is not needed if ρ < 0.4

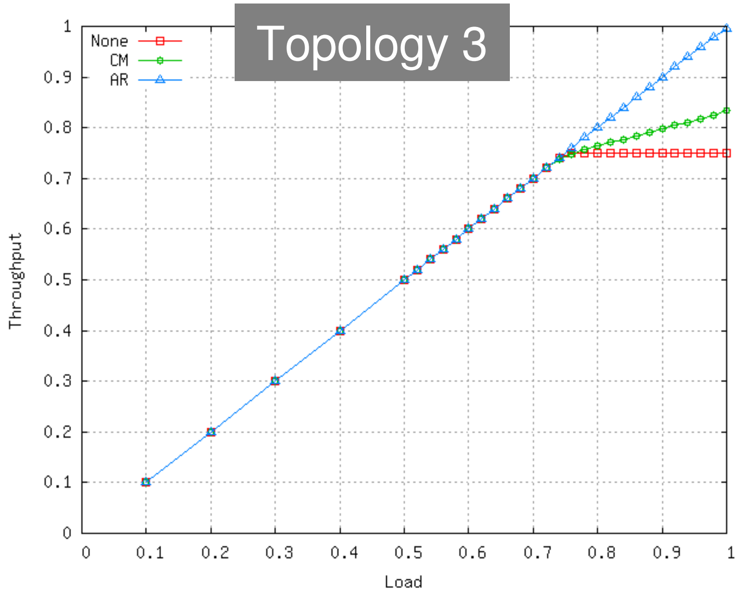
Uniform Bernoulli traffic

Throughput vs. load

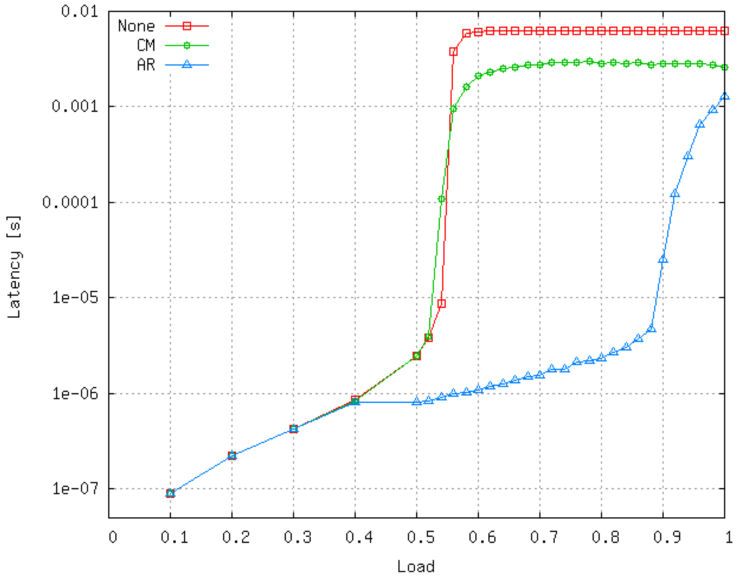
Topology 1

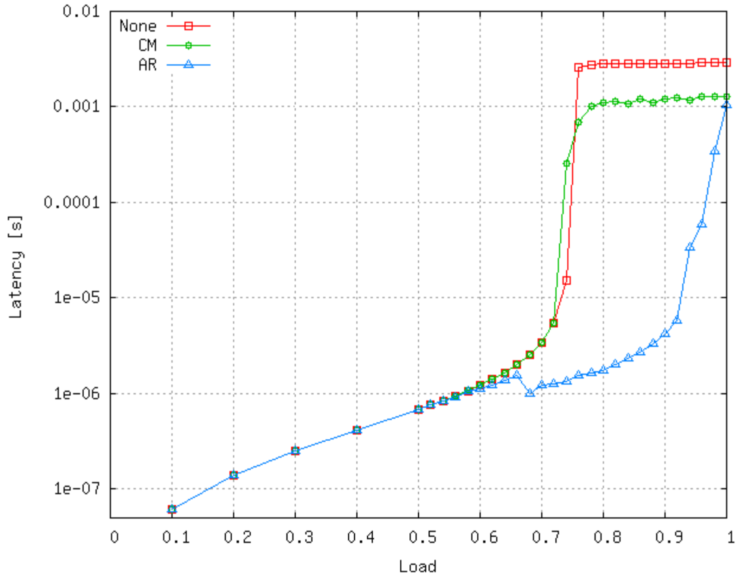


Topology 3



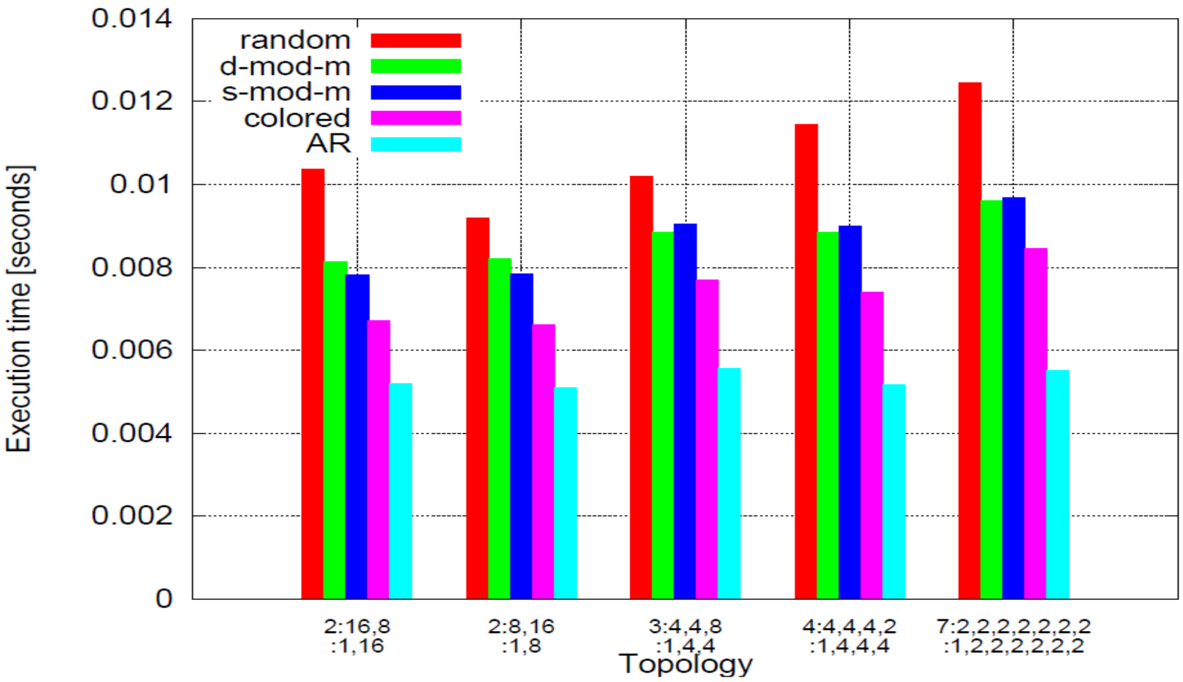
Latency vs. load



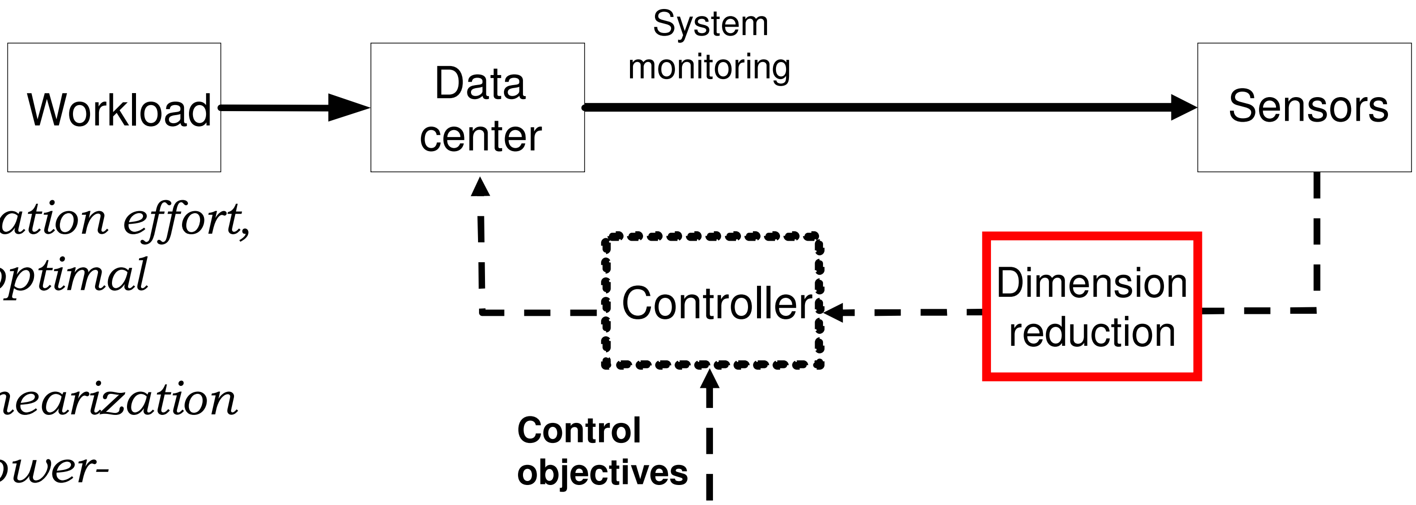


Practical Impact, Benchmarking of Adaptive Routing

- Conjugate Gradient (CG) application from the NAS Parallel Benchmarks (NPB)
- Test run on a *real* 128-node HPC machine to record the trace
- HPC traces files replayed on the VENUS simulator



Analytical Modeling: Online Feedback



- *to reduce the simulation effort, reformulate as an optimal feedback control*
- *apply piece-wise linearization*
- *include the load/power-dependent lag*

$$\dot{x}(t) = A(t)x(t) + B(t)u(t - \tau^{f}(t)) + D(t)w(t)$$
$$y(t) = C(t)x(t - \tau_{tot}(t))$$

- The ultimate goal is a resilient datacenter, that adapts its performance to the workload
  - Feedback required for adaptivity
  - No DC-scale system model yet
- The problem has large dimensionality (10s to 100s)
  - Previous work on dimension reduction for IT infrastructure monitoring
  - Finding simpler models with the least loss of information

Conclusions

- DC benchmarking: everybody wants it, nobody has it...

- DCs and Clouds lack
  - accurate system-level models,
  - rigorous holistic benchmark methodologies
  - agreement on metrics and SLAs ?
  - there's no MPI equivalent for commercial apps (do we need one?)
  - access to operational DCs
    - any volunteers?...☺

- Transaction tracing in DCs remains an open challenge
  - new methods or a breakthrough are needed!

- Simulation and analytics are complementary
  - hard non-linearities (hotspot, server/power overload) need simulation
  - exploring all the sim. dimensions, or finding the worst-case pattern, is not practically feasible via brute force sims ➔ analytics can help

- Scalable simulation environments are tried&true in HPC… no DC/Cloud?
- Can we instrument your datacenter… ? ☺ {That's all, thanks!}