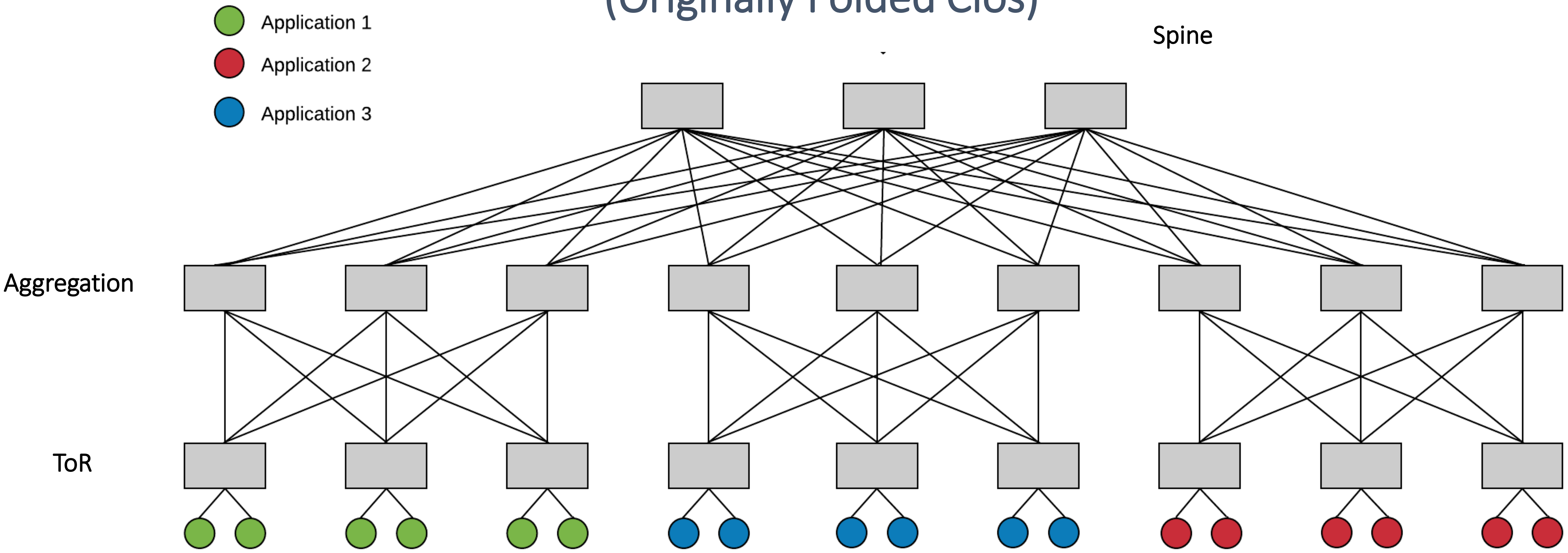
Bandwidth Steering in HPC Using Silicon Nanophotonics

George Michelogiannakis[1], Yiwen Shen[2], Min Yee Teh[2], Xiang Meng[2], Benjamin Aivazi[2], Taylor Groves[1], John Shalf[1], Madeleine Glick[2], Manya Ghobadi[4], Larry Dennison[3], Keren Bergman[2]

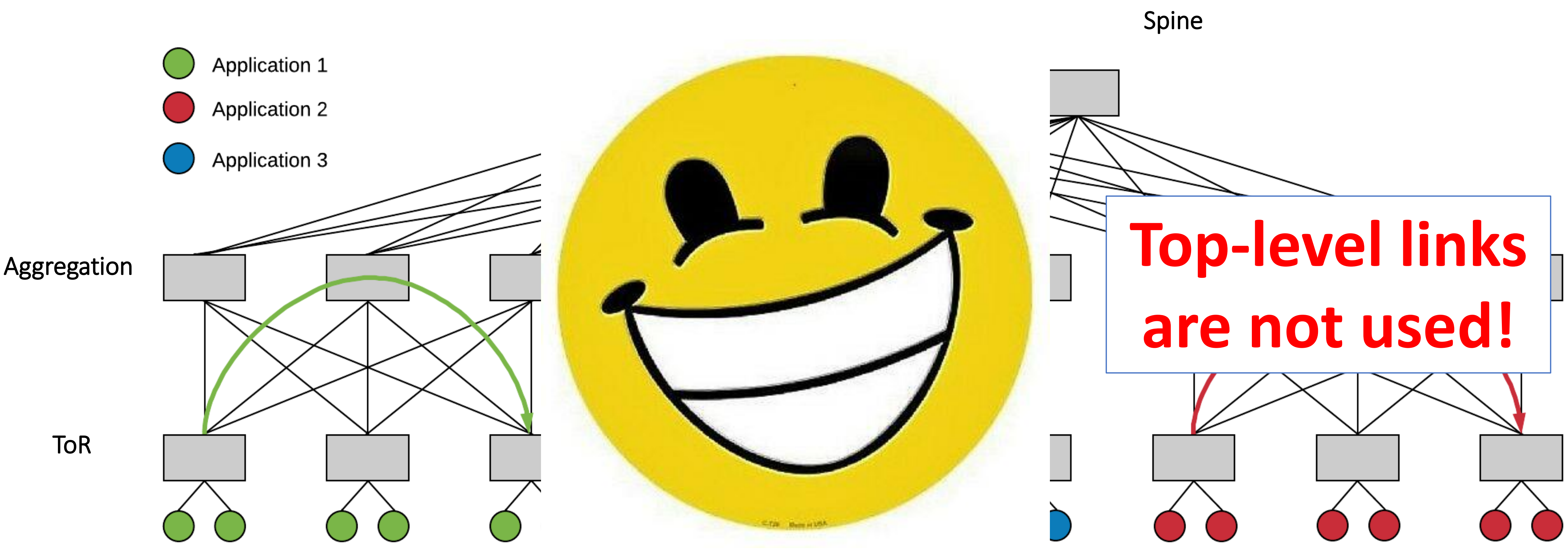
[1]Lawrence Berkeley National Laboratory, [2]Columbia University, [3]NVIDIA, [4]MIT

Standard Static ('Vanilla') Fat Tree Architecture (Originally Folded Clos)

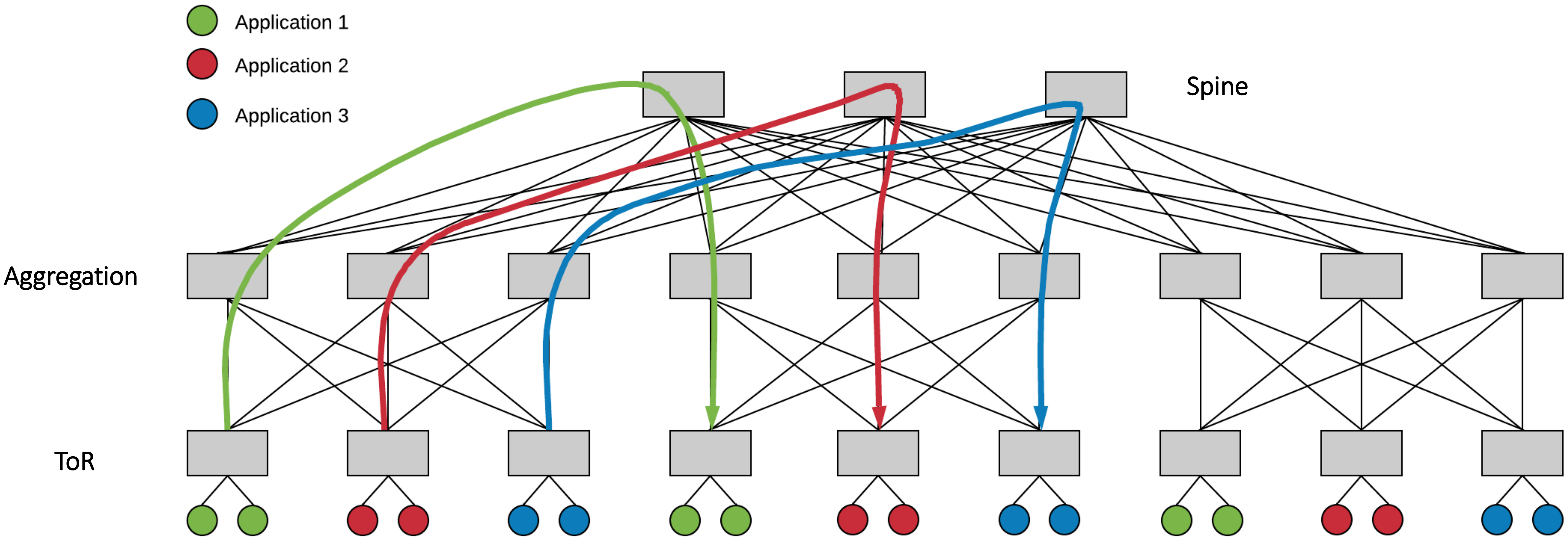


- Fat tree is a proxy for hierarchical topologies

Favorable Task Placement



Worst-Case Placement



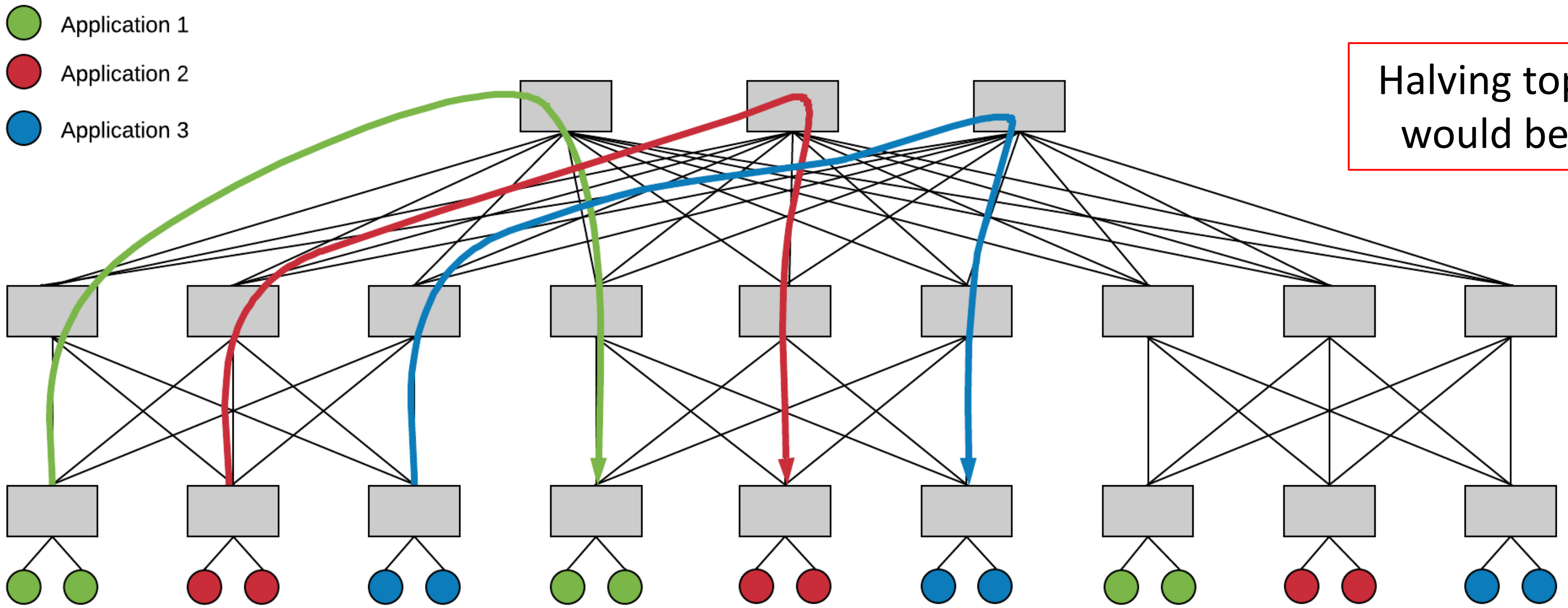
- Inter-pod traffic needs to traverse (+ hops) through spine to connect between pods

The Problem: Bandwidth Tapering

Bandwidth tapering removes higher-level expensive bandwidth

Examples: Facebook 4x oversubscription [1], Microsoft up to 5.3x [2]

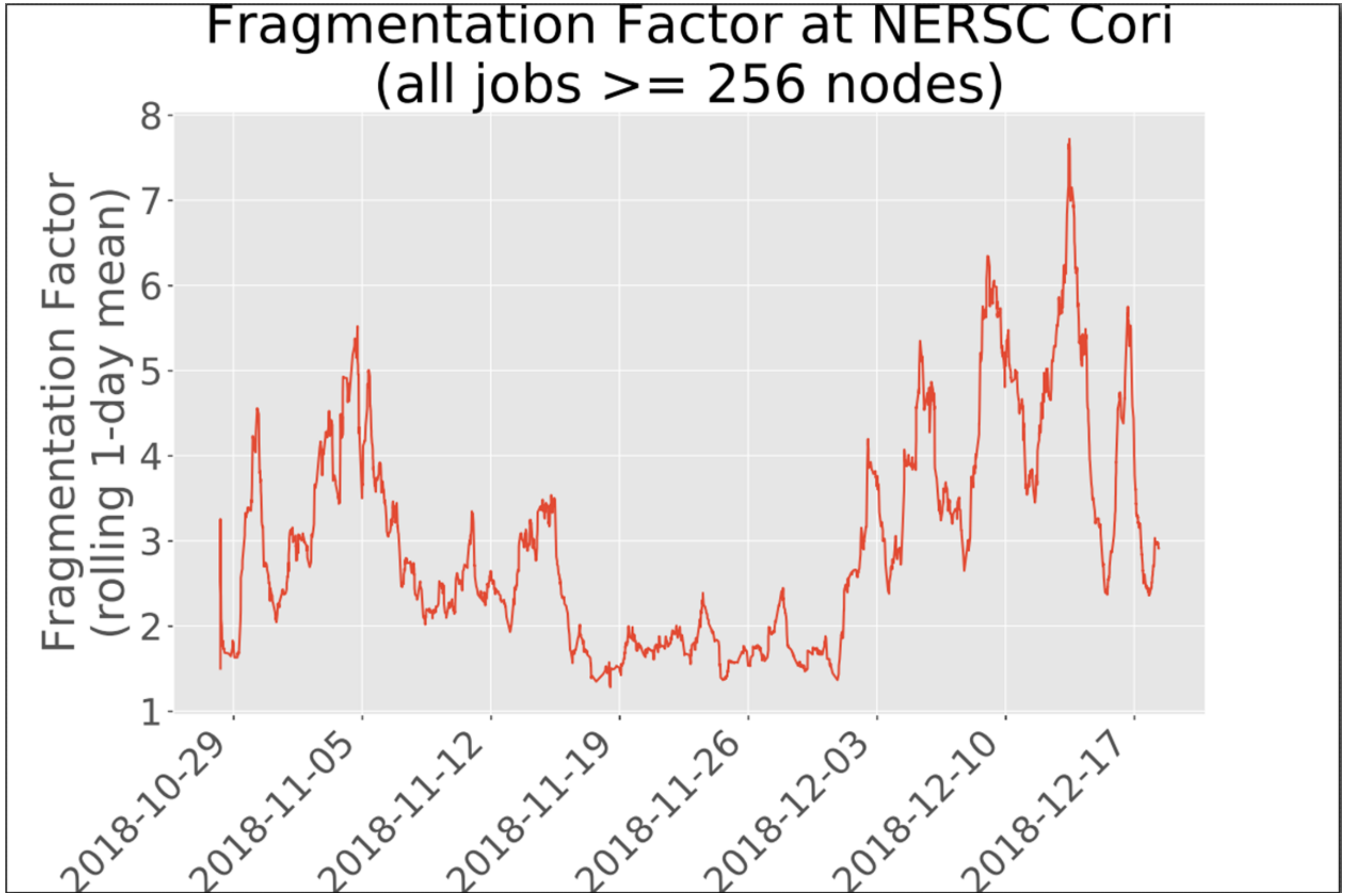
If all traffic uses high level, congestion forms [2]



Halving top-level cables would be 2x tapering

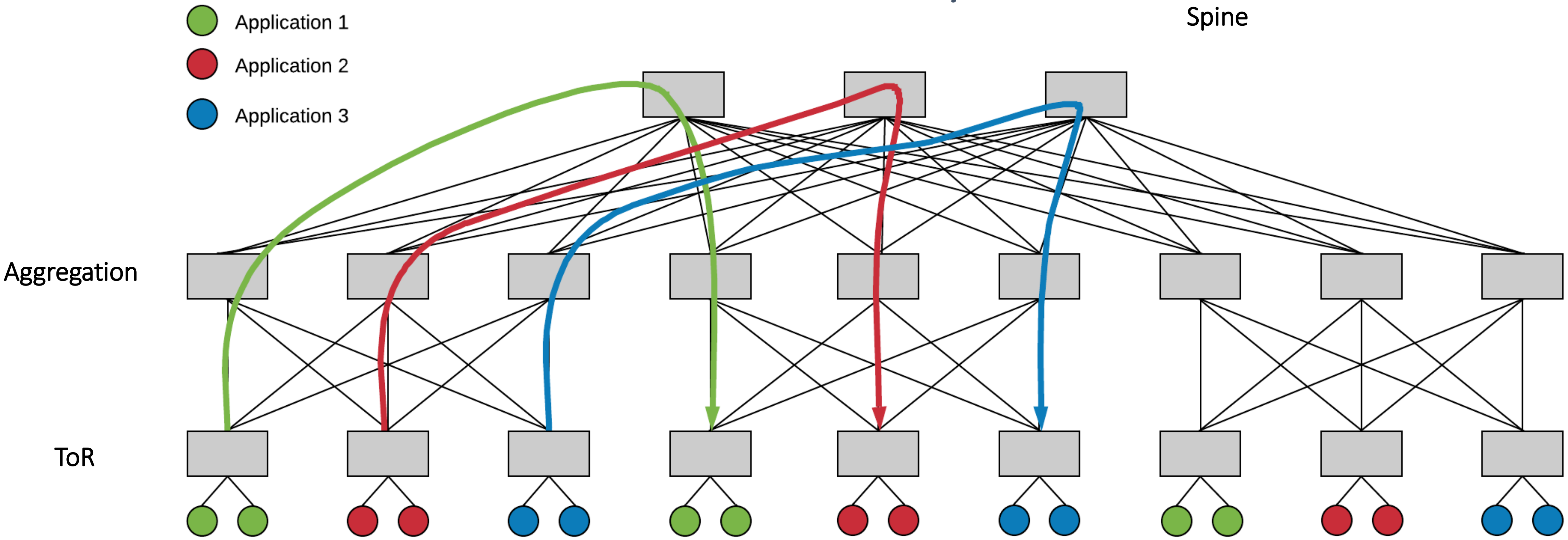
[1] Andreyev, Alexei et al, "Introducing data center fabric, the next-generation Facebook data center network", Facebook engineering, 2014
[2] Chatzieleftheriou, Andromachi et al, "Larry: Practical Network Reconfigurability in the Data Center", NSDI, 2018

Fragmentation in NERSC’s Cori

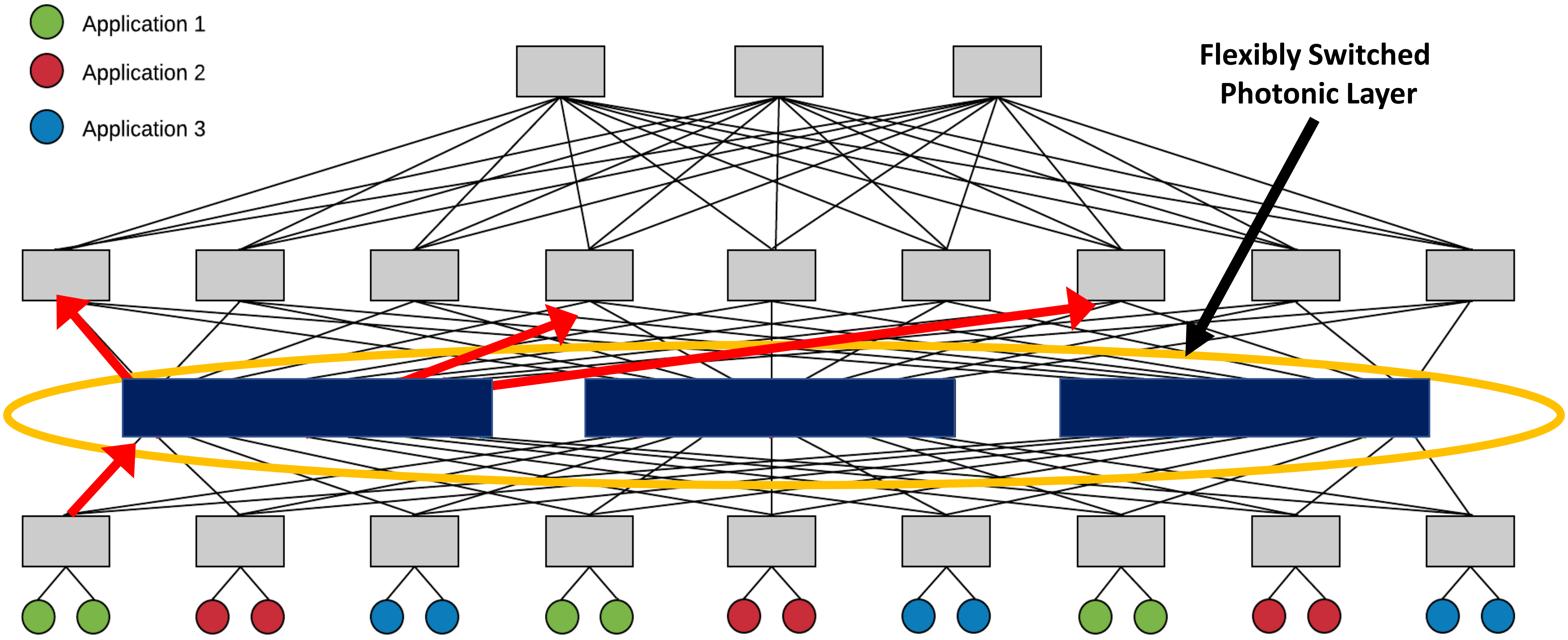


- Metric: Observed number of Aries (Dragonfly) groups that an application spans, divided by the smallest possible number of groups that the application would fit in.

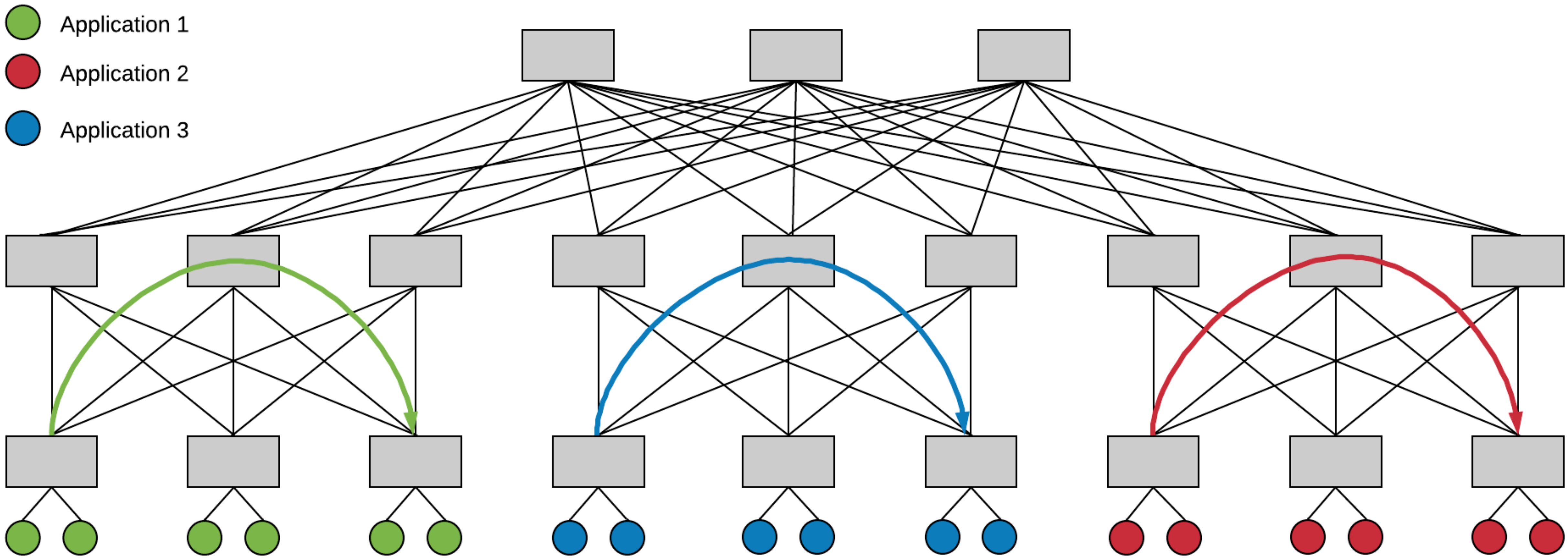
Problem Statement: Recover Locality by Changing Topology Connectivity



Flexible Fat Tree: Insertion of SiP Switches – Bandwidth Steering

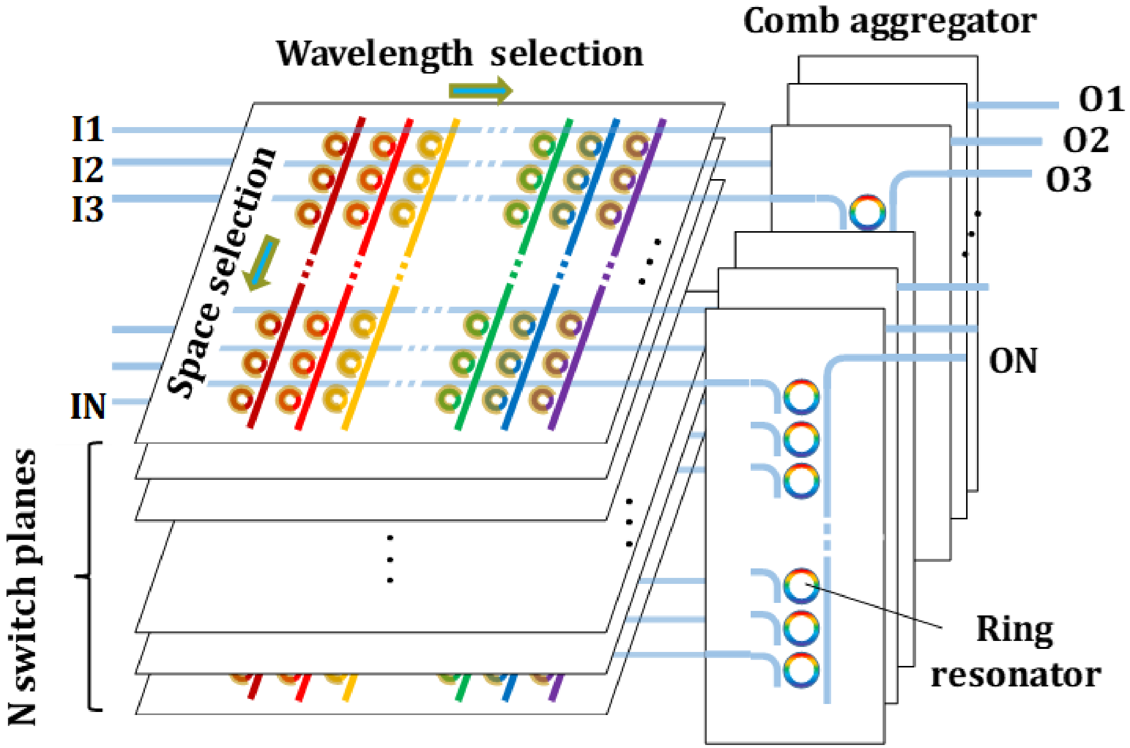


Flexible Fat Tree: Direct Connectivity with Bandwidth Steering

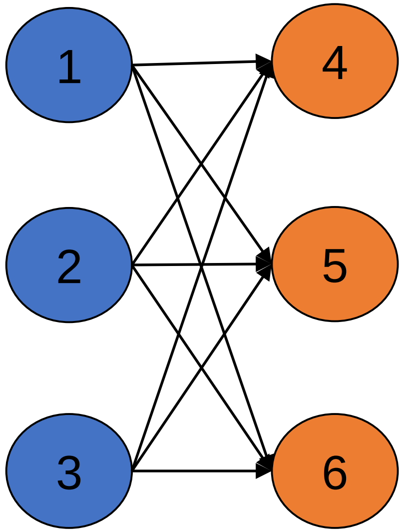


Why Optical Switches Efficiently Steer B/W

- Negligible dynamic power and latency for traversal
- Orders of magnitude lower static power than modern electronic switches
- However:
  - We avoid consecutive hops in the optical domain to avoid optical loss
  - No buffering inside optical switches. They need to be pre-configured (circuit switching)



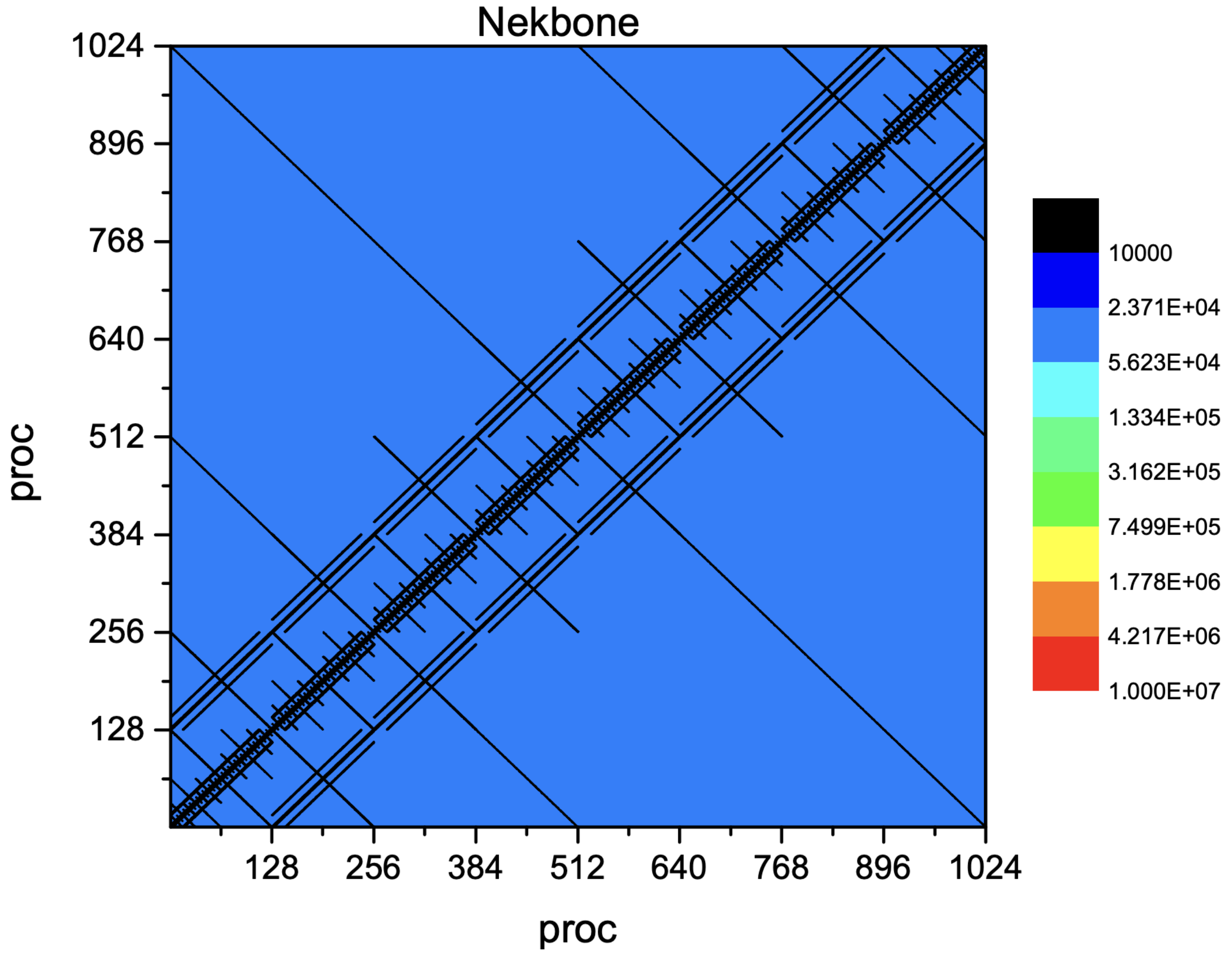
Reconfiguration Algorithm



- Traffic estimation or observation
  - A multi-node job starts every 17 seconds
  - PINE switches reconfigure in ~20usec
  - Commercial switches every few msec
- Algorithm is heuristic. Optimal solution is NP-Complete
  - Iteratively solve in each optical switch a maximum-weight matching problem
  - On-line update matching weights, considering already established links between pod pairs
  - Scalable: **O(kr^4)**
    - *k* is # of SiP switches in network. *r* is optical radix and tends to be small

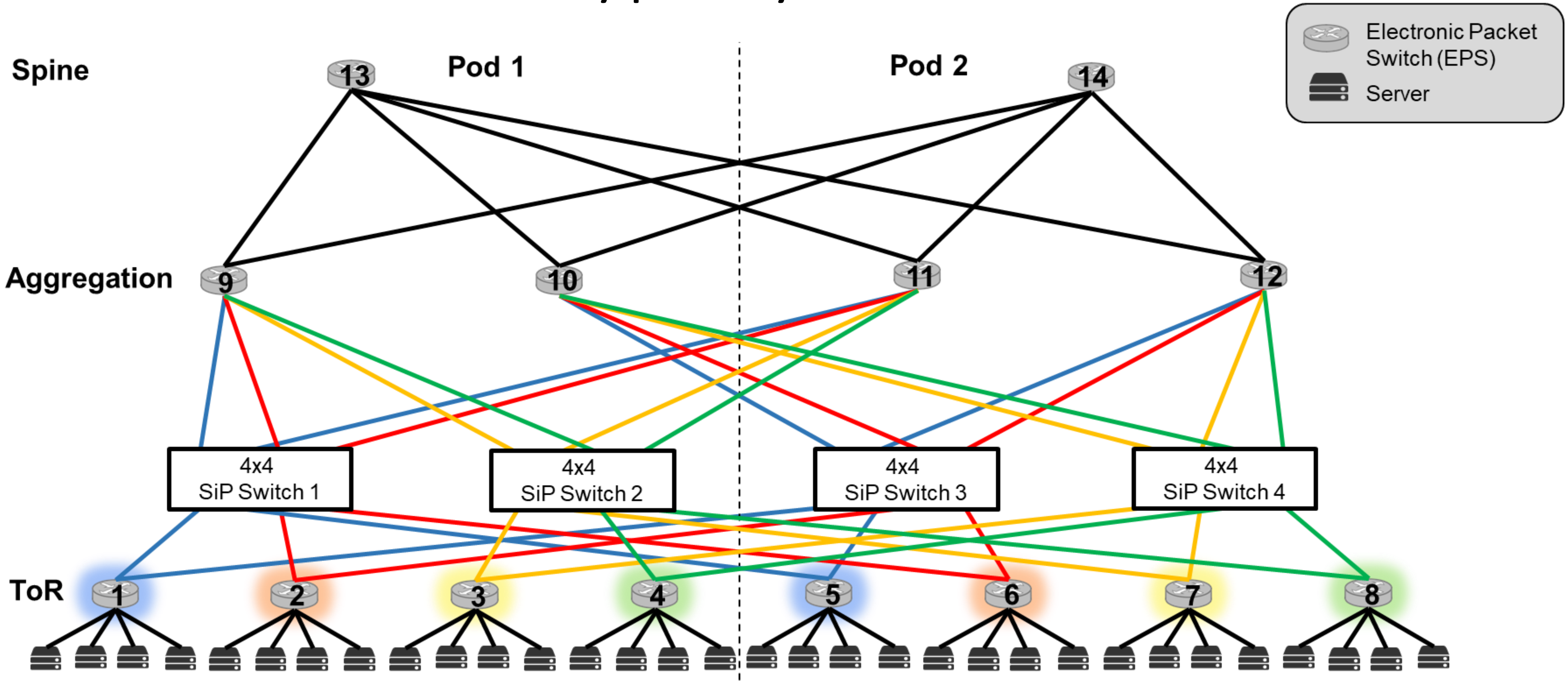
Traffic Patterns Persist

- Applications may go through phases, but the dominant pattern persists throughout



DOE exascale miniapps: https://portal.nersc.gov/project/CAL/cesar.htm

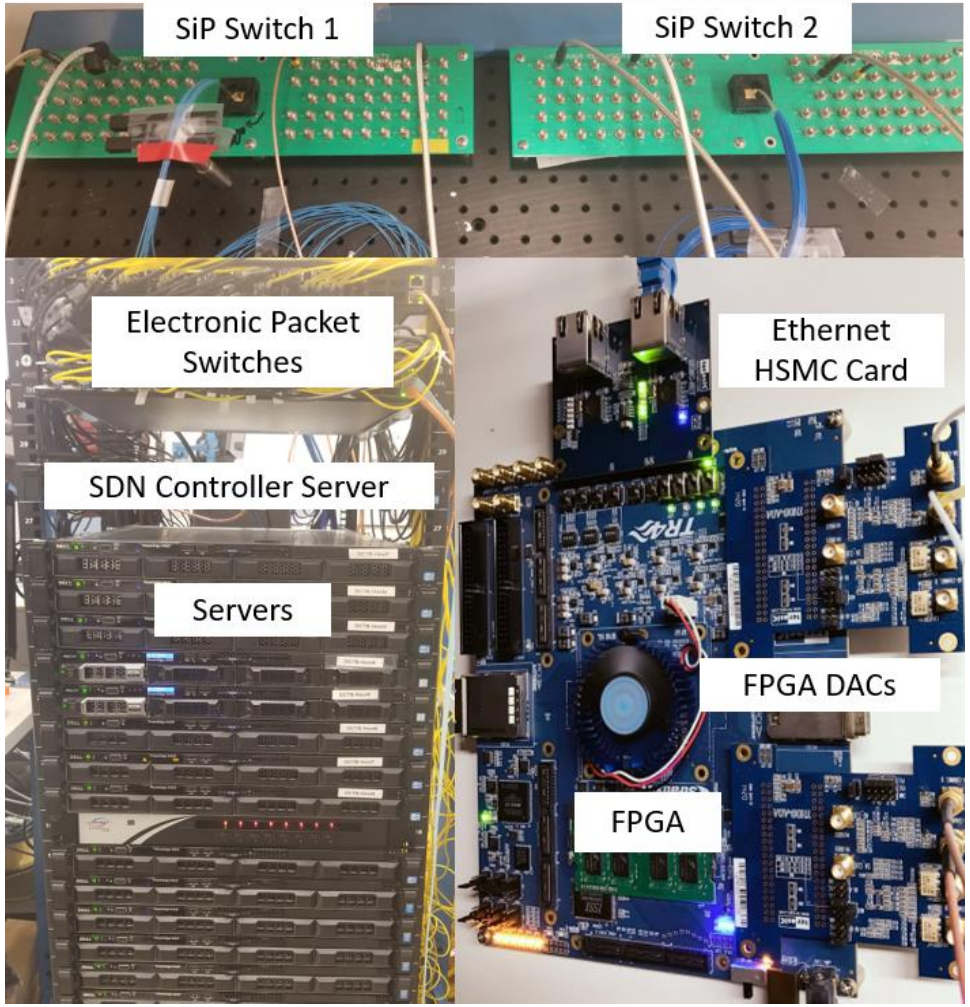
PINE Prototype System Testbed

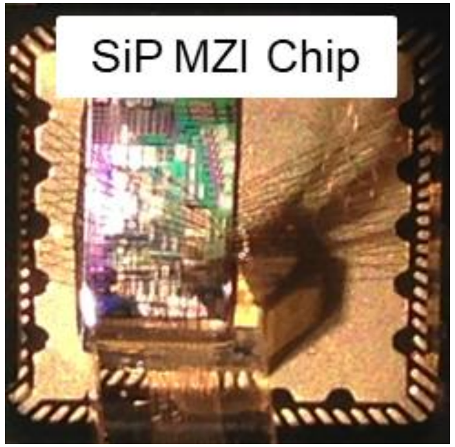


- 32 compute nodes composed of VMs on 16 servers, with 10G NICs
- Electronic virtually partitioned from two OpenFlow PICA8 Ethernet packet switches (48 10G SFP+ ports)

arpa·e CHANGING WHAT'S POSSIBLE PINE

PINE Bandwidth Steering Architecture – Fat Tree Topology: Prototype System Testbed



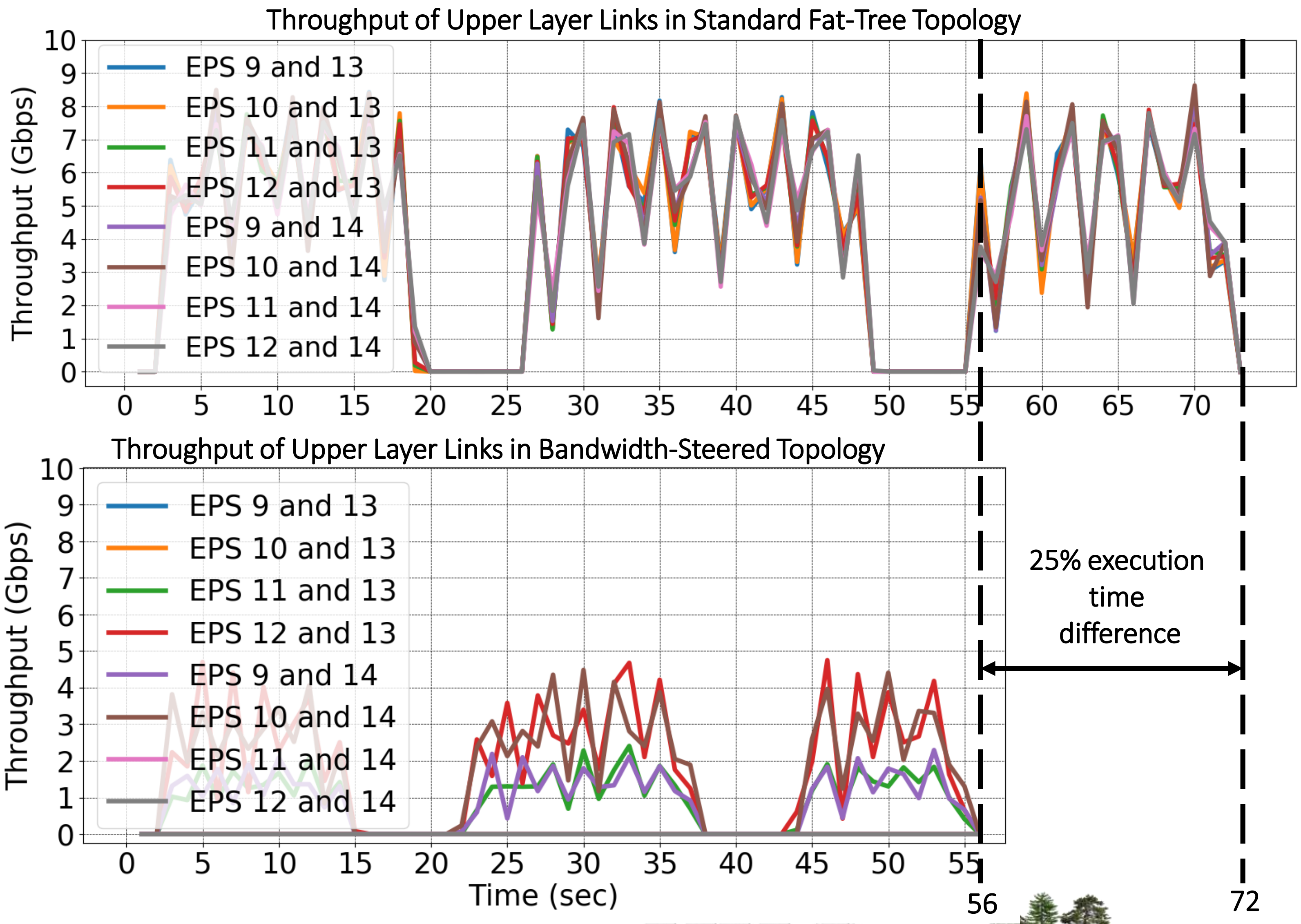


Standard Fat Tree Versus Steered Fat Tree

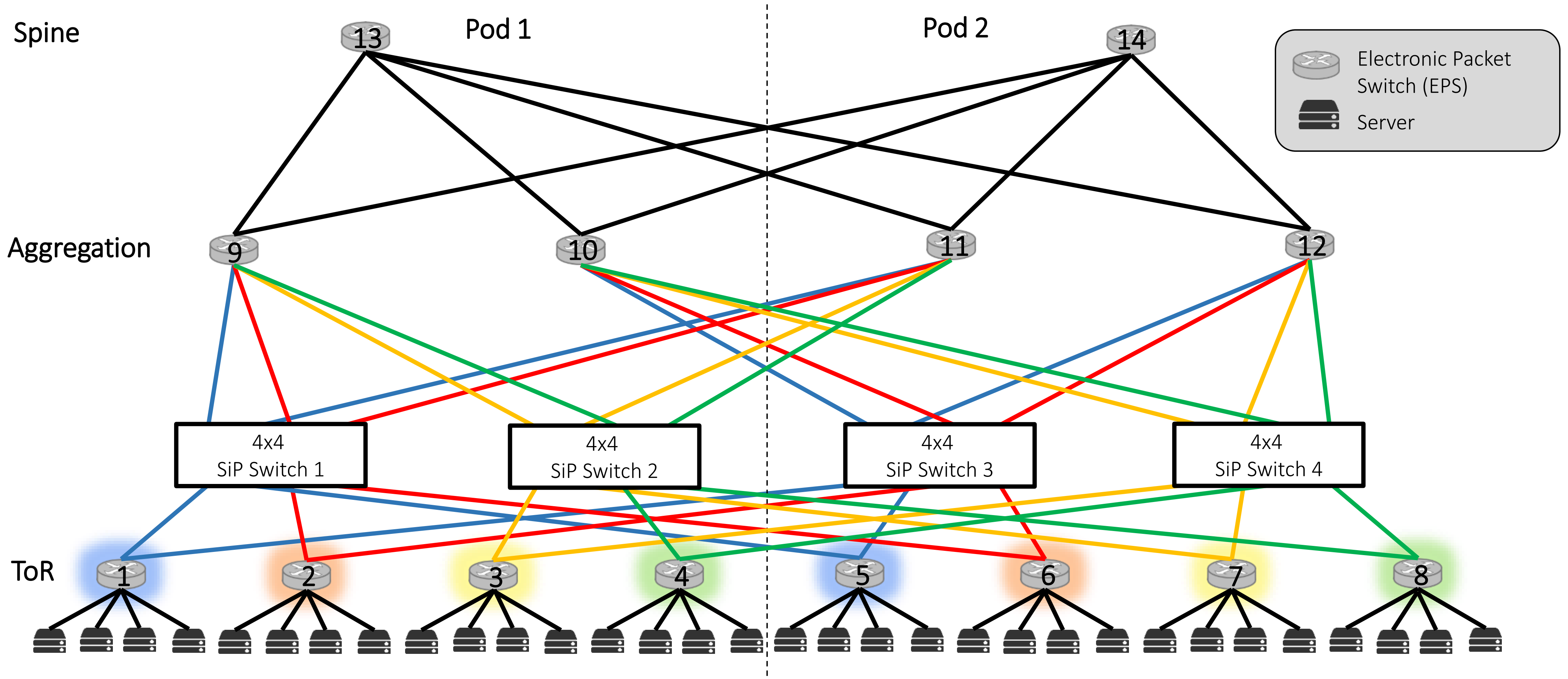
- Operating skeletonized Gyrokinetic Toroidal Code (GTC) application with MPI

- Standard Fat Tree: all upper layer links used

- Flexible PINE Fat Tree: Only 4 upper layer links used



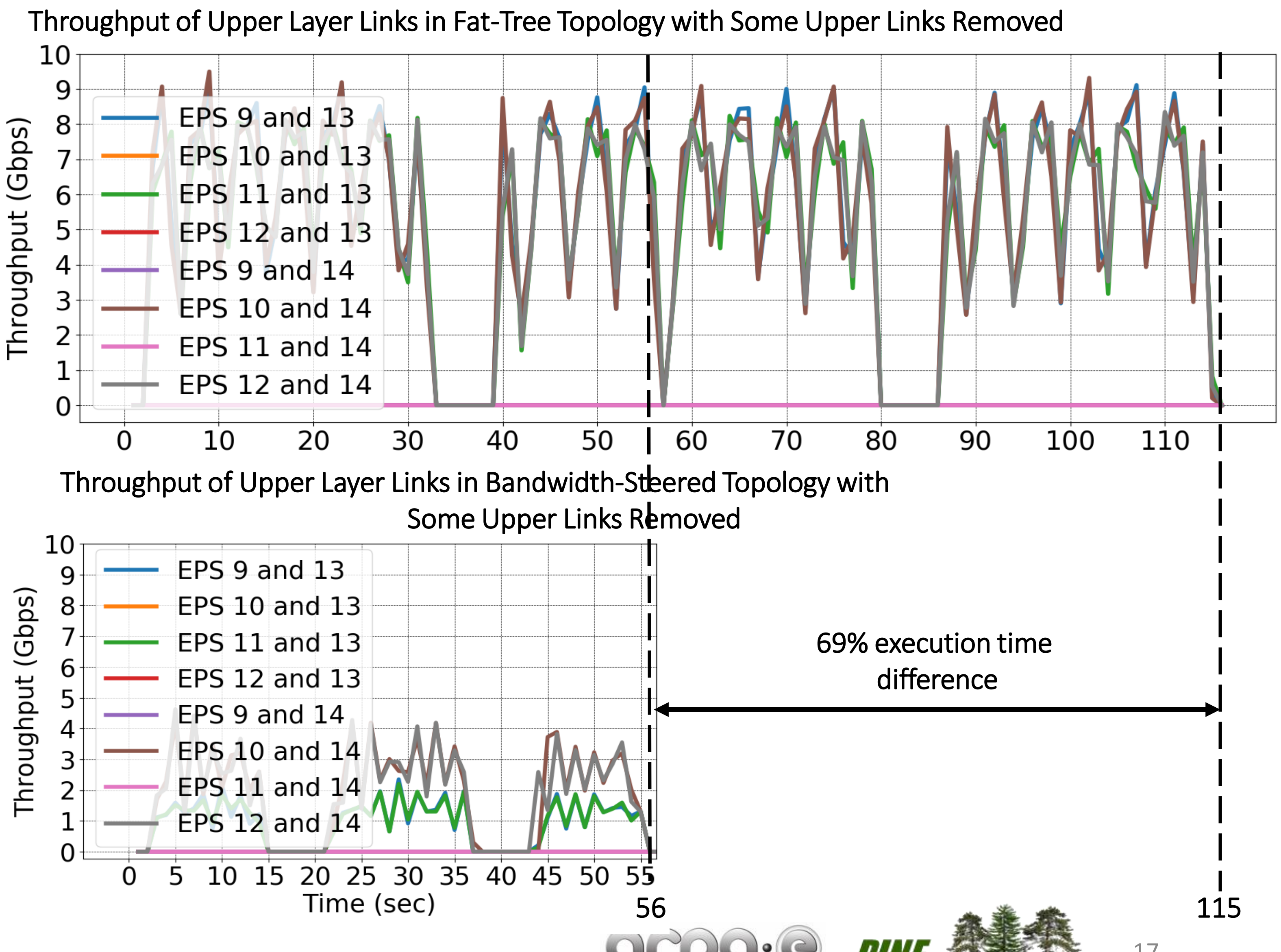
2x Oversubscription (B/W Tapering)



- Remove spine layer links to reduce energy consumption

Flexible Fat Tree Unaffected by Tapering

- Standard Fat Tree: all remaining upper layer links congested
- Flexible Fat Tree: <u>unaffected</u> by removal of links, remains at 56 sec runtime
- **69% faster execution**



System-Scale Evaluation Methodology

- Booksim simulator
- Minimal-path routing with oblivious per-packet load balancing
- Network size and radix chosen for each trace
- Randomize placement to simulate fragmentation

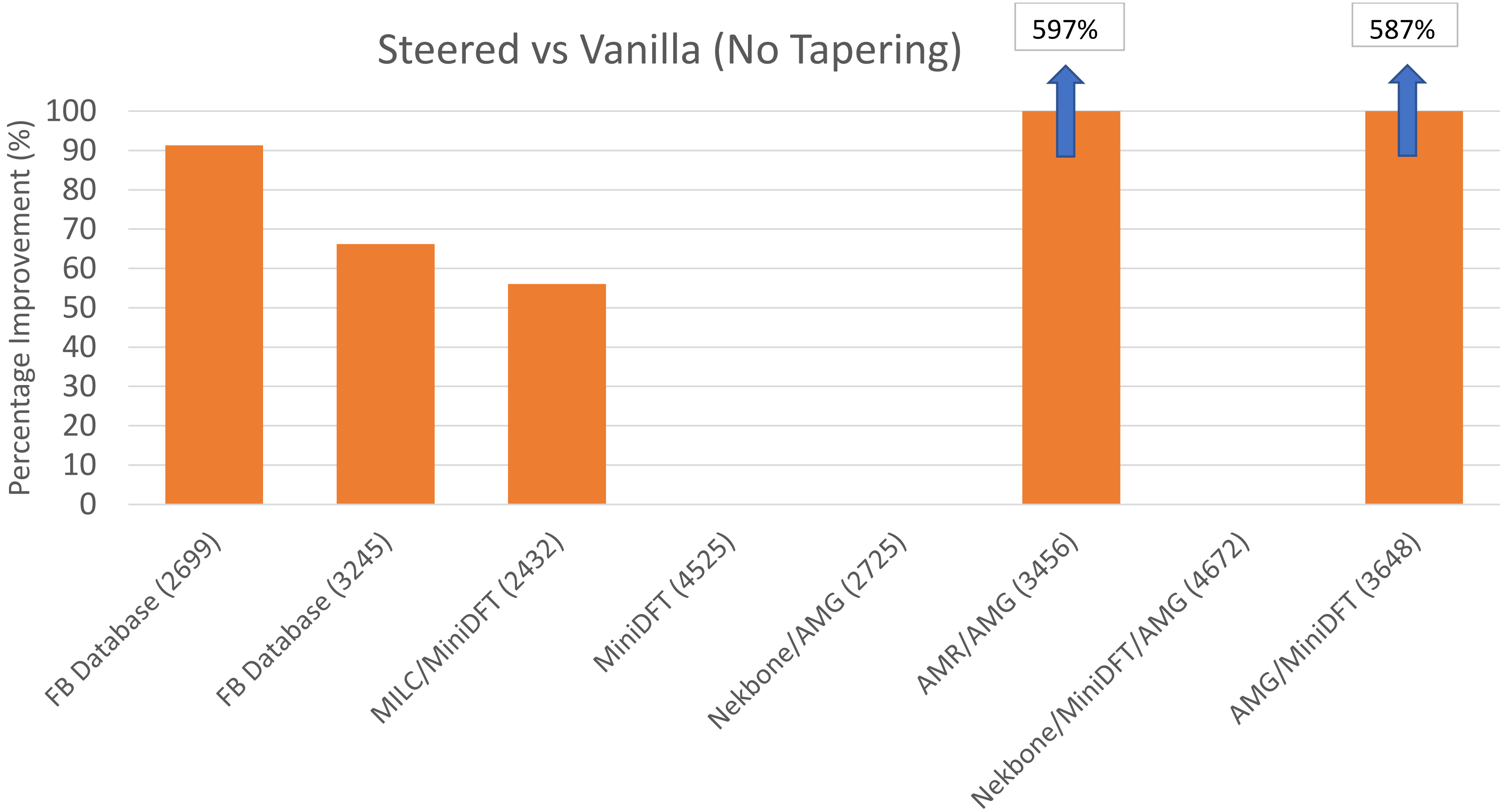
- 36x36 Mellanox switches and active optical cables
  - 16x16 SiP optical switches

CHANGING WHAT'S POSSIBLE PINE

System-Scale Evaluation Traces

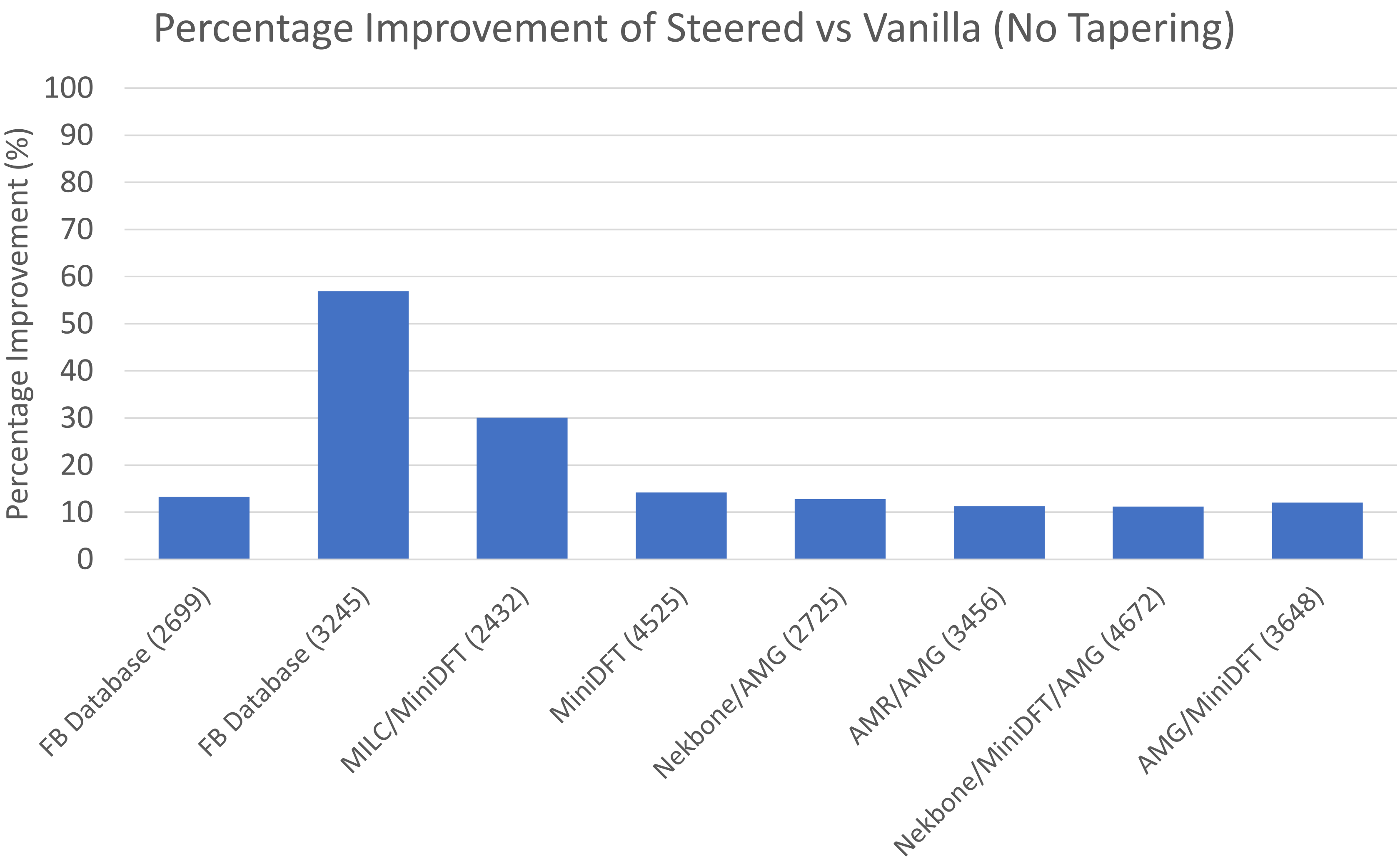
Application	Algorithm
Facebook	Production-level database pod
MiniDFT	Plane-wave density functional theory (DFT)
MILC	4D stencil with nearest-neighbor traffic
Nekbone	Poison equation using conjugate gradient iteration
AMG	Algebraic multigrid solver (AMG)
AMR	Adaptive mesh refinement (AMR)

Transactions per Second (Throughput)



- Average 1.7x improvement
  - Large spread
- Fat tree in the baseline:
  - Operated past saturation at times
  - Load balancing was not perfect

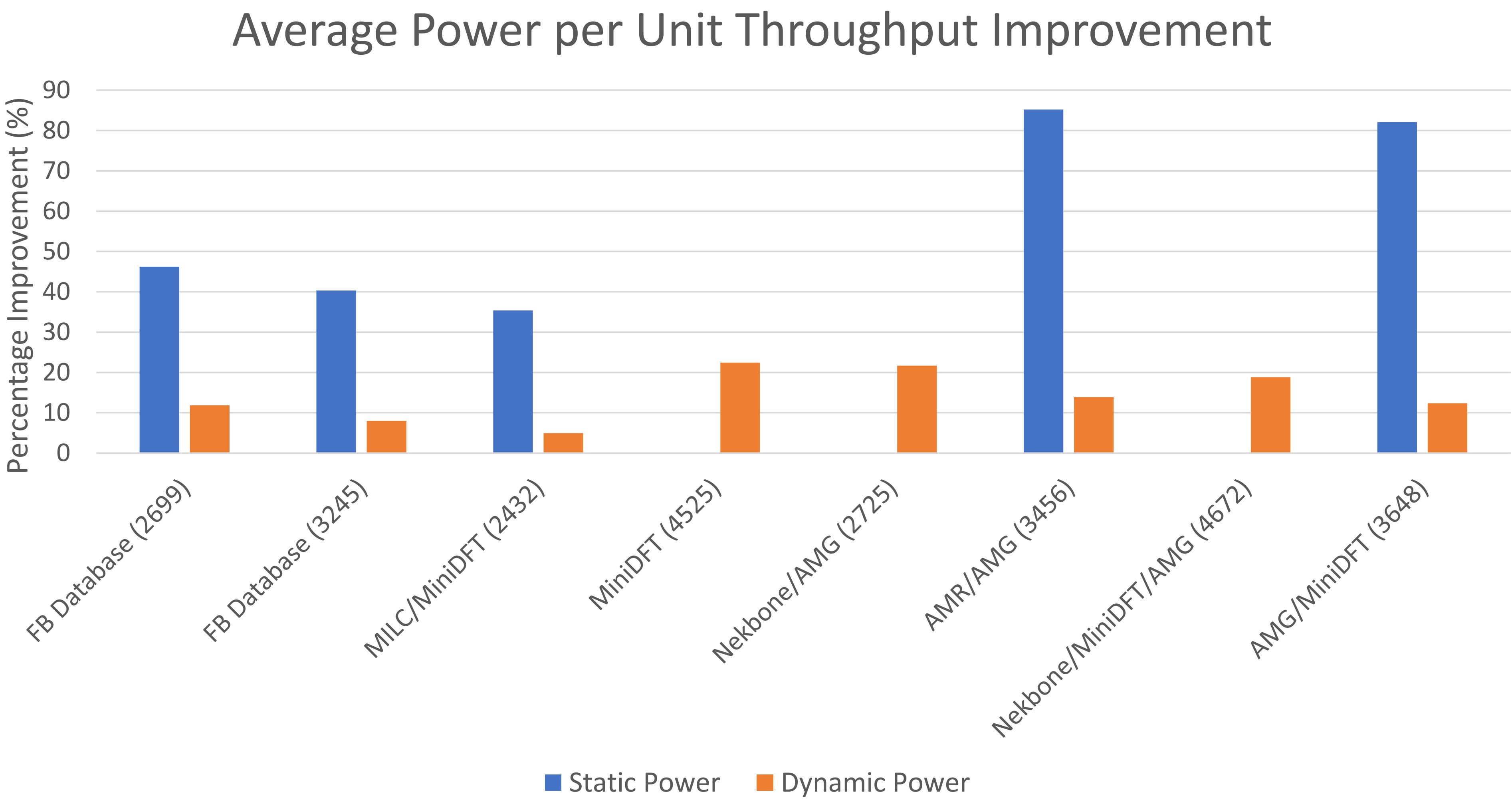
Network Average Latency



- Average 20% improvement
- Two reasons:
  - Lower hop count
  - Lower congestion in some cases

CHANGING WHAT'S POSSIBLE

Power Consumption



- Average 36% static and 14% dynamic

SiP Optical Switch Radix

- If we reduce SiP optical switch radix 8x8, throughput drops appreciably in only two traces
  - Plenty of locality to recover with lower-radix switches

- Much of related work used expensive high-radix optical switches

CHANGING WHAT'S POSSIBLE

PINE

Related Work / Task Migration

- Previous studies show a disjoint optical network for heavy traffic, optimize for metrics other than tapering and fragmentation, or provide reconfiguration but require large-radix optical switches

- Task migration can take seconds to complete [1]. Our optical switches reconfigure in microseconds. Electronic switches in milliseconds

[1] Chao Wang et al, "Proactive Process-level Live Migration in HPC Environments". SC 2008

Summary

- Bandwidth steering reconstructs locality lost from system fragmentation and reduces higher-level link utilization
  - Therefore, can aggressively taper higher-layers with no performance penalty
- SiP optical switches efficiently change the connectivity of lower layers to match the traffic pattern
- 36% less static and 14% less dynamic power per unit throughput
  - Also 69% faster execution in our testbed

Questions?