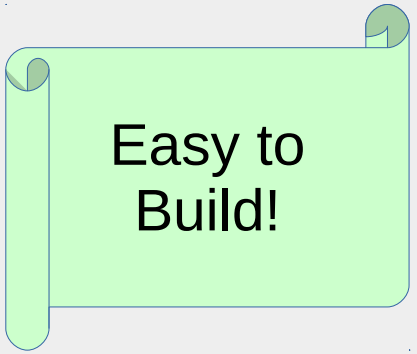
A Methodology for Characterizing the Correspondence Between Real and Proxy Applications

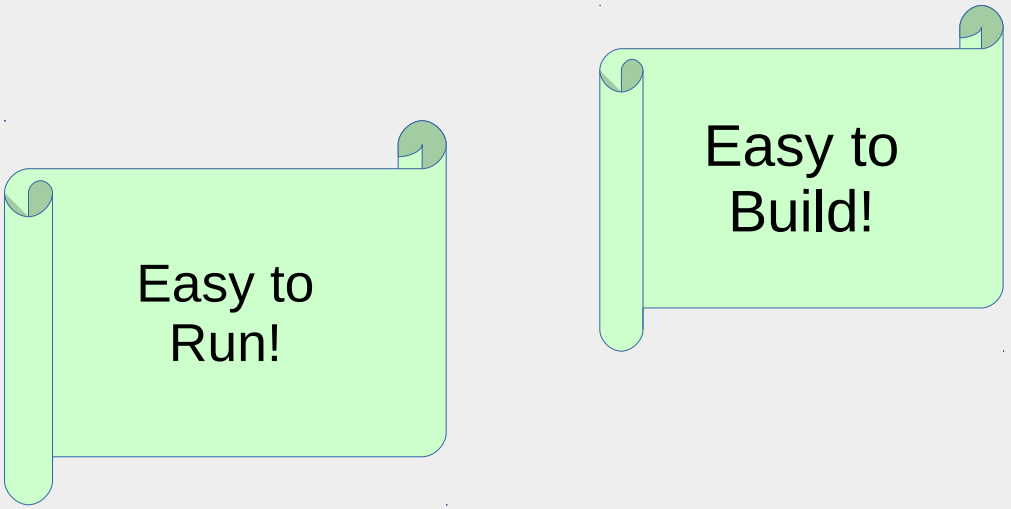
Omar Aaziz (SNL)
Jeanine Cook (SNL)
Jonathan Cook (NMSU)
Tanner Juedeman (NMSU)
David Richards (LLNL)
Courtenay Vaughan (SNL)

Proxy Apps are Wonderful!

Proxy Apps are Wonderful!



Proxy Apps are Wonderful!



Proxy Apps are Wonderful!

Easy to Build!

Easy to Run!

Self-Monitoring!

Proxy Apps are Wonderful!

Easy to Modify!

Easy to Run!

Easy to Build!

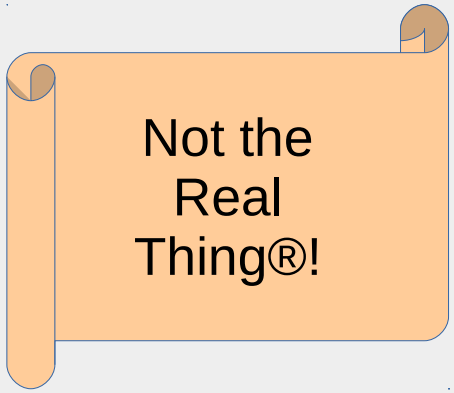
Self-Monitoring!

Explore Ideas!

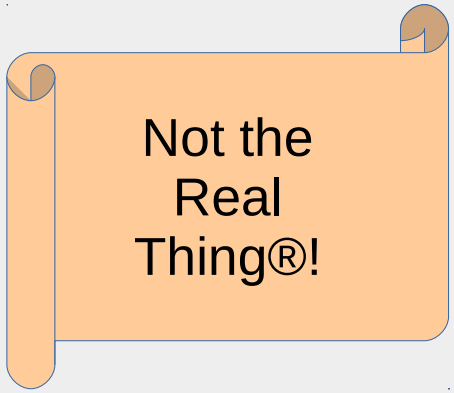
Proxy Apps are Horrible!

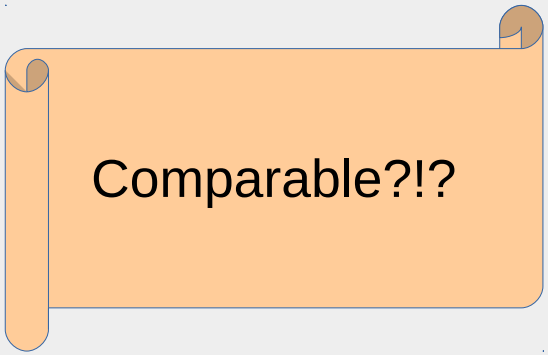
Proxy Apps are just OK

Proxy Apps are just OK



Proxy Apps are just OK





Proxy Apps are just OK

Not the Real Thing®!

Benchmark issues

Comparable?!?

Proxy Apps are just OK

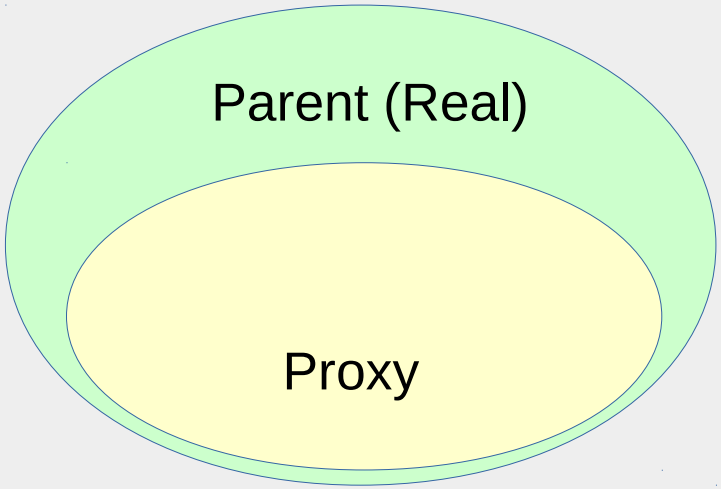
Not the Real Thing®!

Benchmark issues

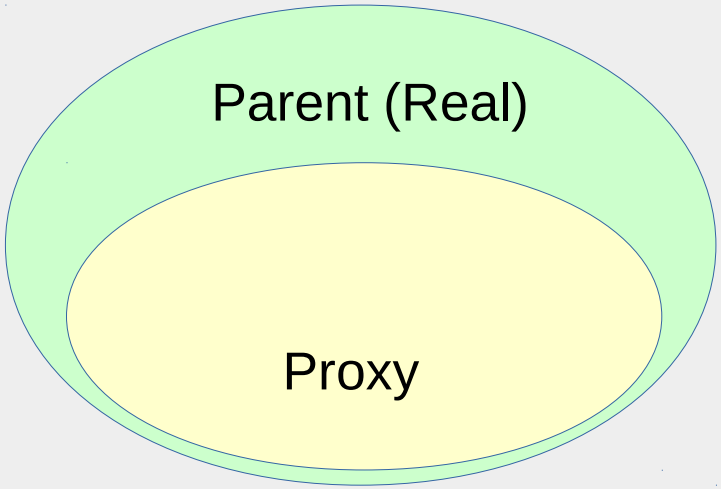
Comparable?!?

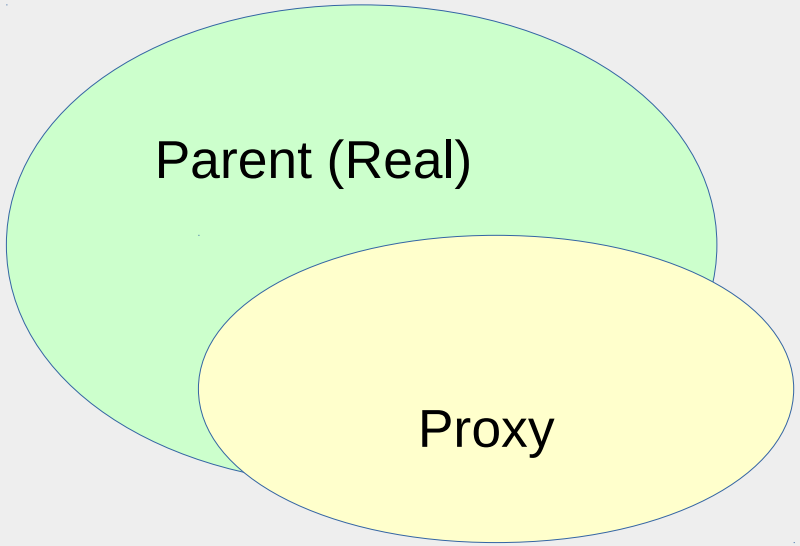
(How) Do they match their parent(s)?

Do Proxies Match the Real Thing?

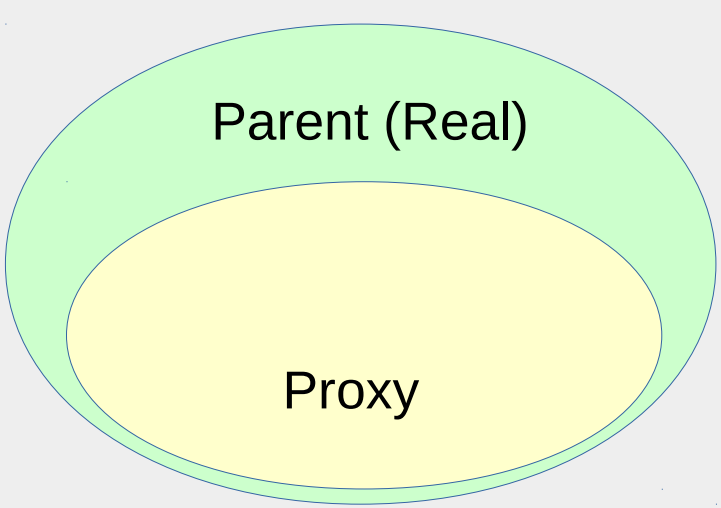


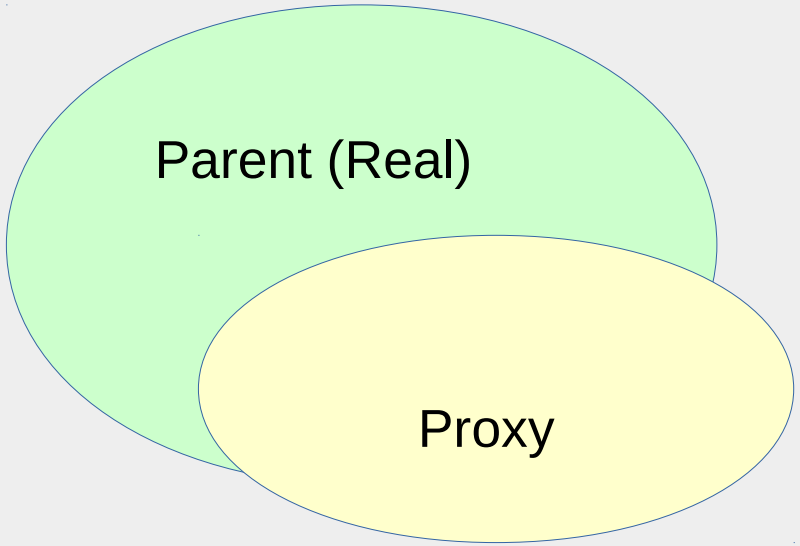
Do Proxies Match the Real Thing?

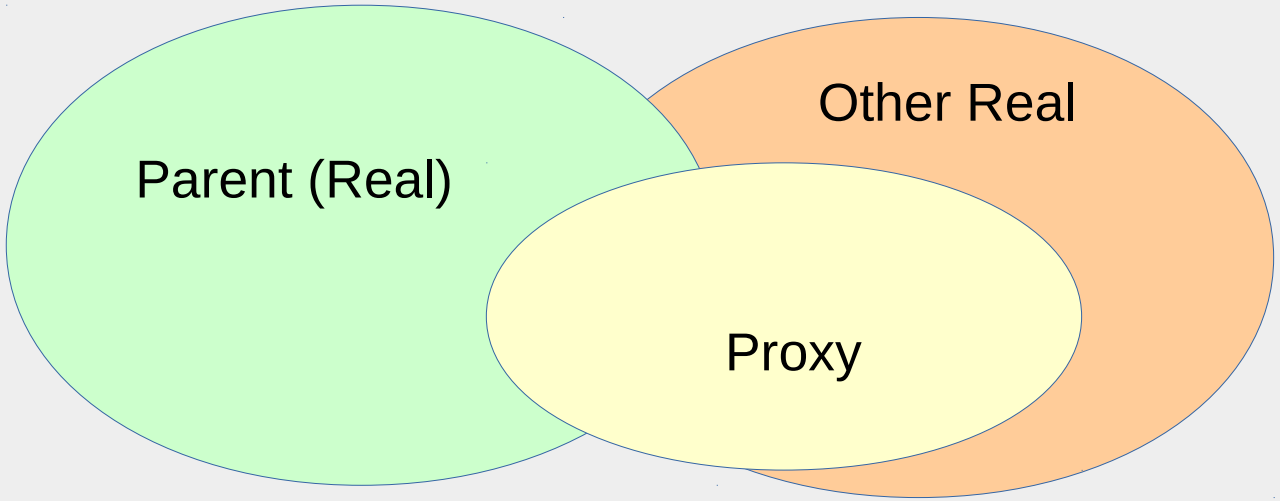




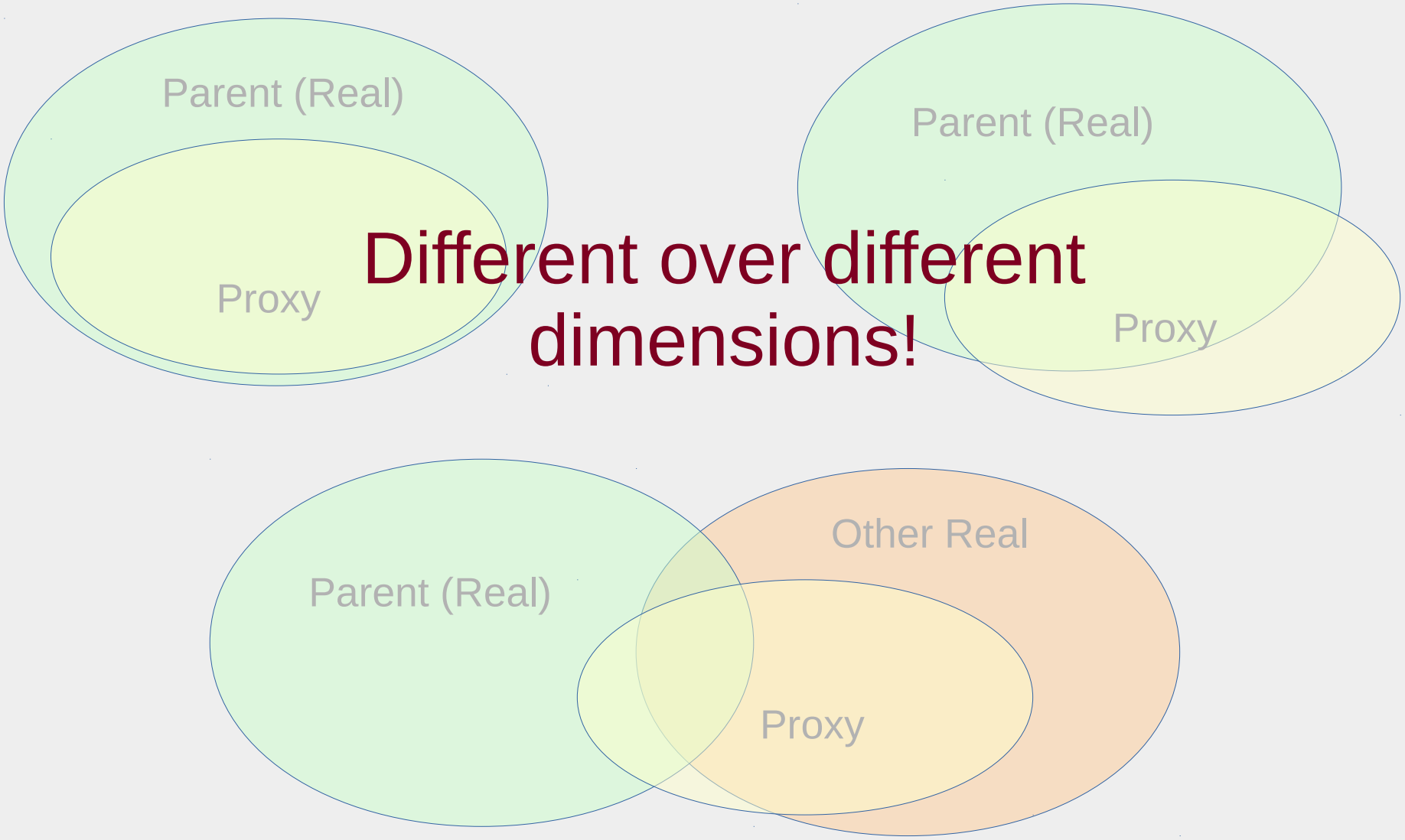
Do Proxies Match the Real Thing?







Do Proxies Match the Real Thing?



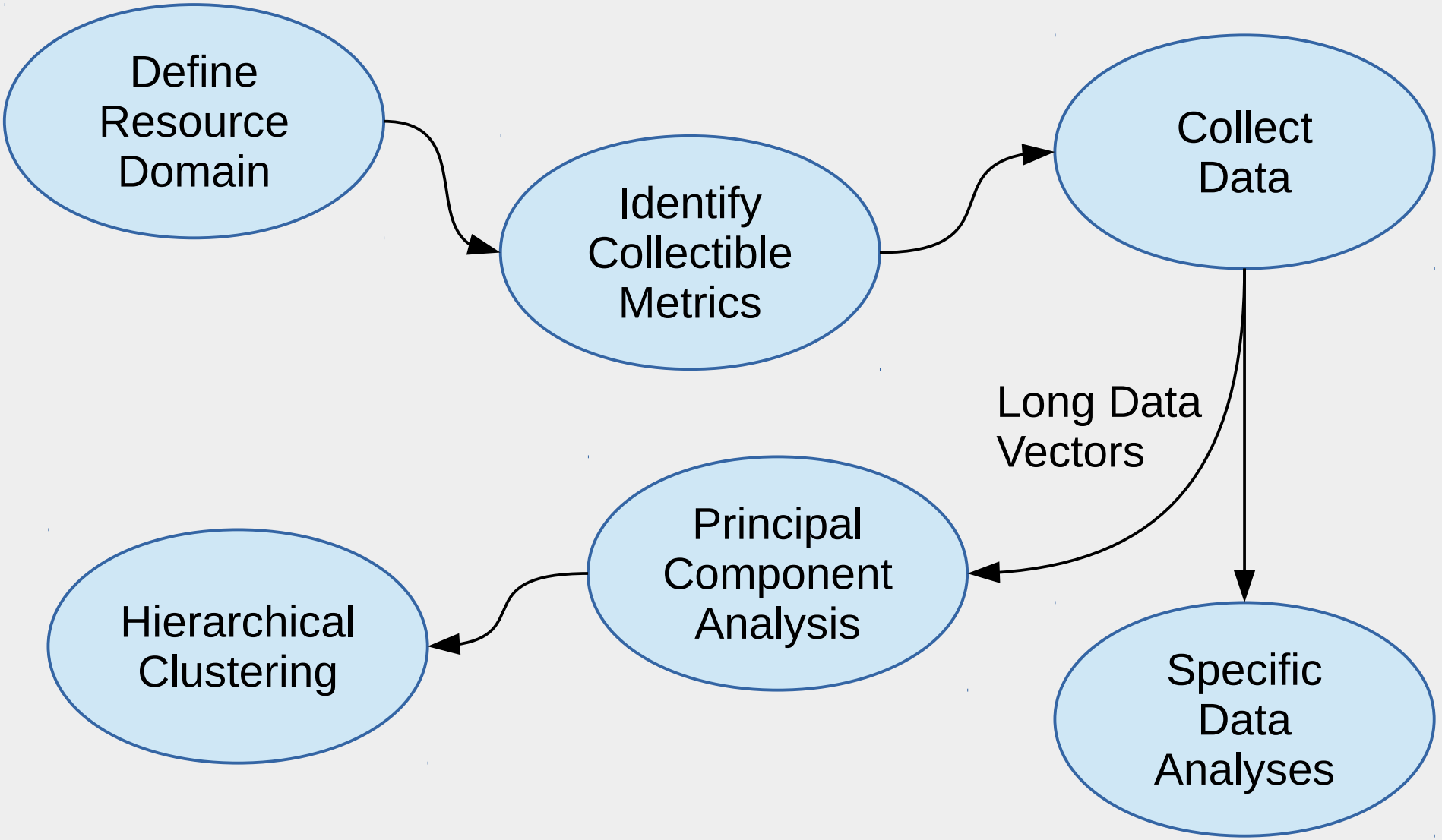
A Methodology

- Define a reusable approach to evaluating parent/proxy correspondences
- Preference towards simplicity
  - But still effective
- Instantiation may be customized
  - For different platforms
  - For different middleware / foundations

Target: Dynamic Behavior

- Goal is to evaluate if proxy exercises the resources similar to the parent
- Measurements should target dynamic behavior
  - Without high perturbation
- Currently: assume that parent and proxy configurations are similar
  - I.e., user knows what they want
  - Build configuration
  - Run configuration

Methodology Flow



FAL, https://commons.wikimedia.org/w/index.php?curid=385145

Dimensions: Resource Domains

- Basic Node
  - Host processors and memory
- Communication
  - Cluster interconnect
- Accelerator
  - GPU, et al.
- Storage I/O
  - Filesystem

Dimensions: Resource Domains

- Basic Node
  - Host processors and memory
- Communication
  - Cluster interconnect
- Accelerator
  - GPU, et al.
- Storage I/O
  - Filesystem

Not much proxy support, or unavailable resources, so did not do

Dimensions: Resource Domains

- Basic Node
  - Host processors and memory
- Communication
  - Cluster interconnect

Some success, but needs improvement

- Accelerator
  - GPU, et al.
- Storage I/O
  - Filesystem

Not much proxy support, or unavailable resources, so did not do

Dimensions: Resource Domains

- Basic Node
  - Host processors and memory
- Communication
  - Cluster interconnect
- Accelerator
  - GPU, et al.
- Storage I/O
  - Filesystem

Good results!

Some success, but needs improvement

Not much proxy support, or unavailable resources, so did not do

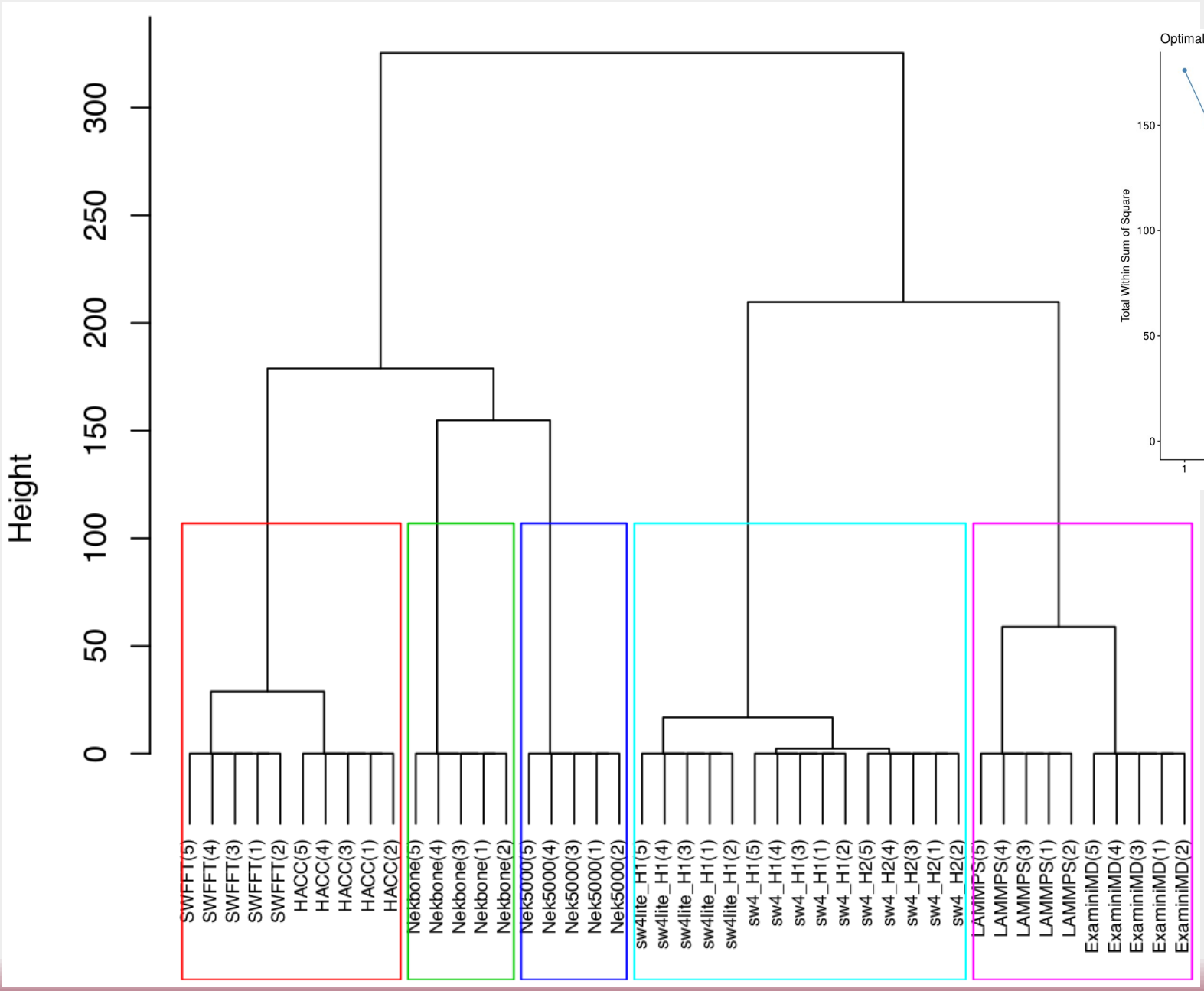
Parents and Proxies

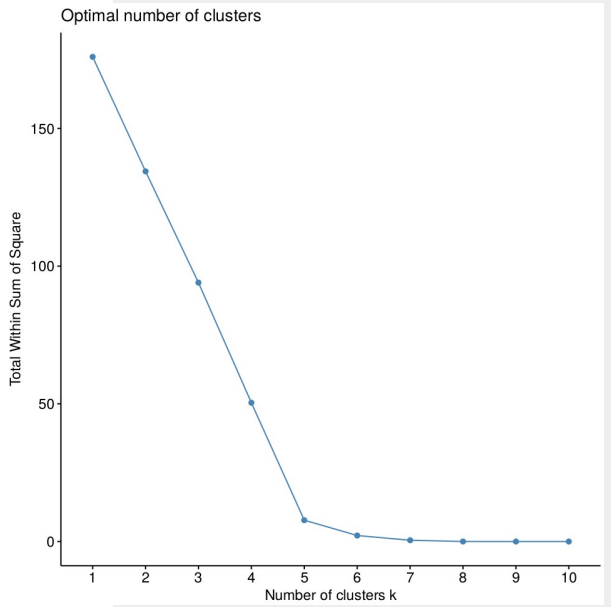
- HACC / SWFFT
- SW4 / SW4lite
- LAMMPS / ExaMiniMD
- Nek5000 / Nekbone

RD: Basic Node

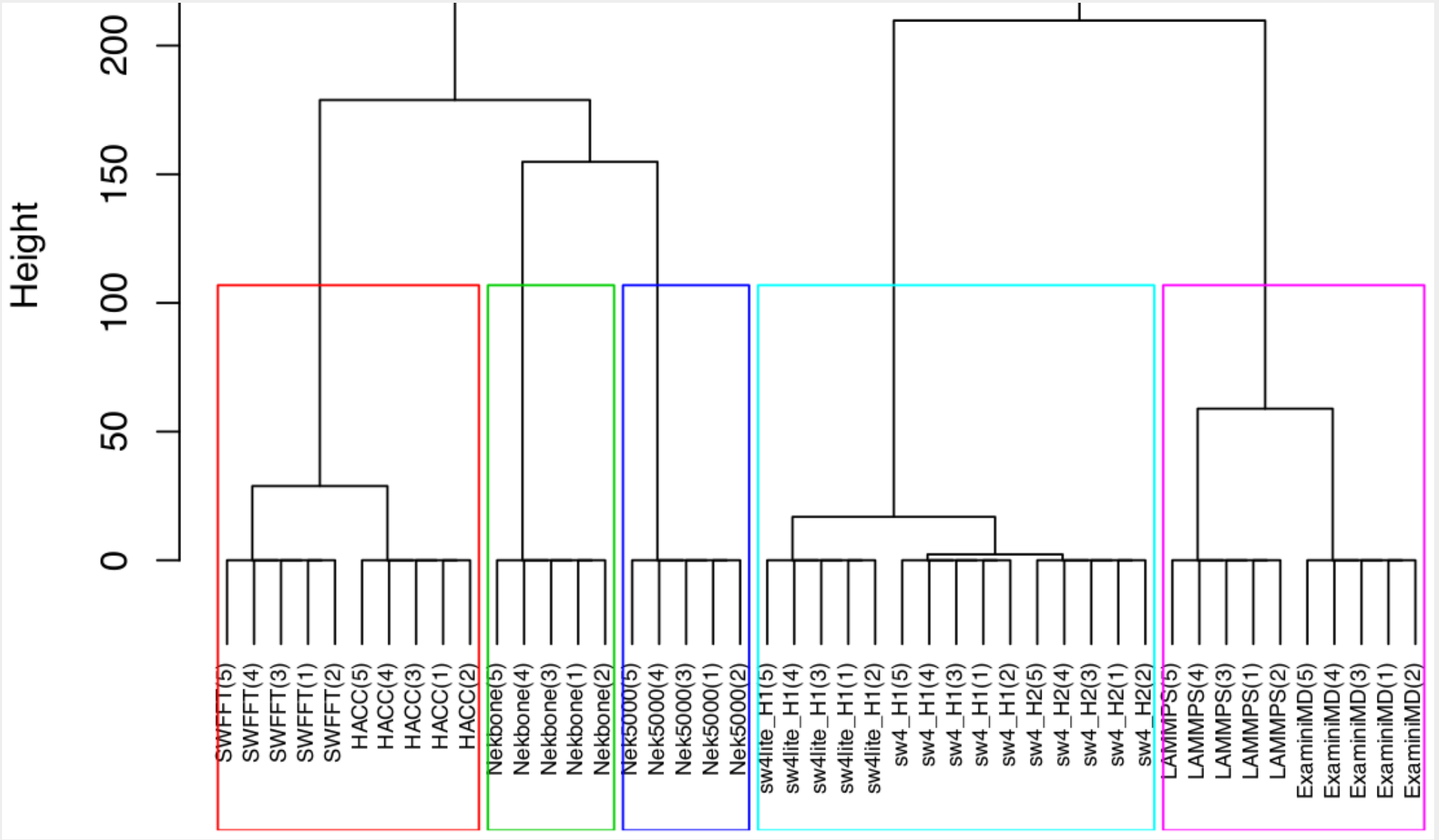
- Used selected metrics over CPU hardware counters
- CPU
  - IPC, UPC, IMIX (5), FLOPS (1-N)
- Memory
  - L1/L2/L3 miss rate, L1/L2/L3 miss ratio, L1-L2-L3 bandwidth
- Vector Size: 22 (*8) on Broadwell, 15 (*8) on Haswell
  - Data collected from rank 0 and 7 random ranks

BW: Basic Node Clustering

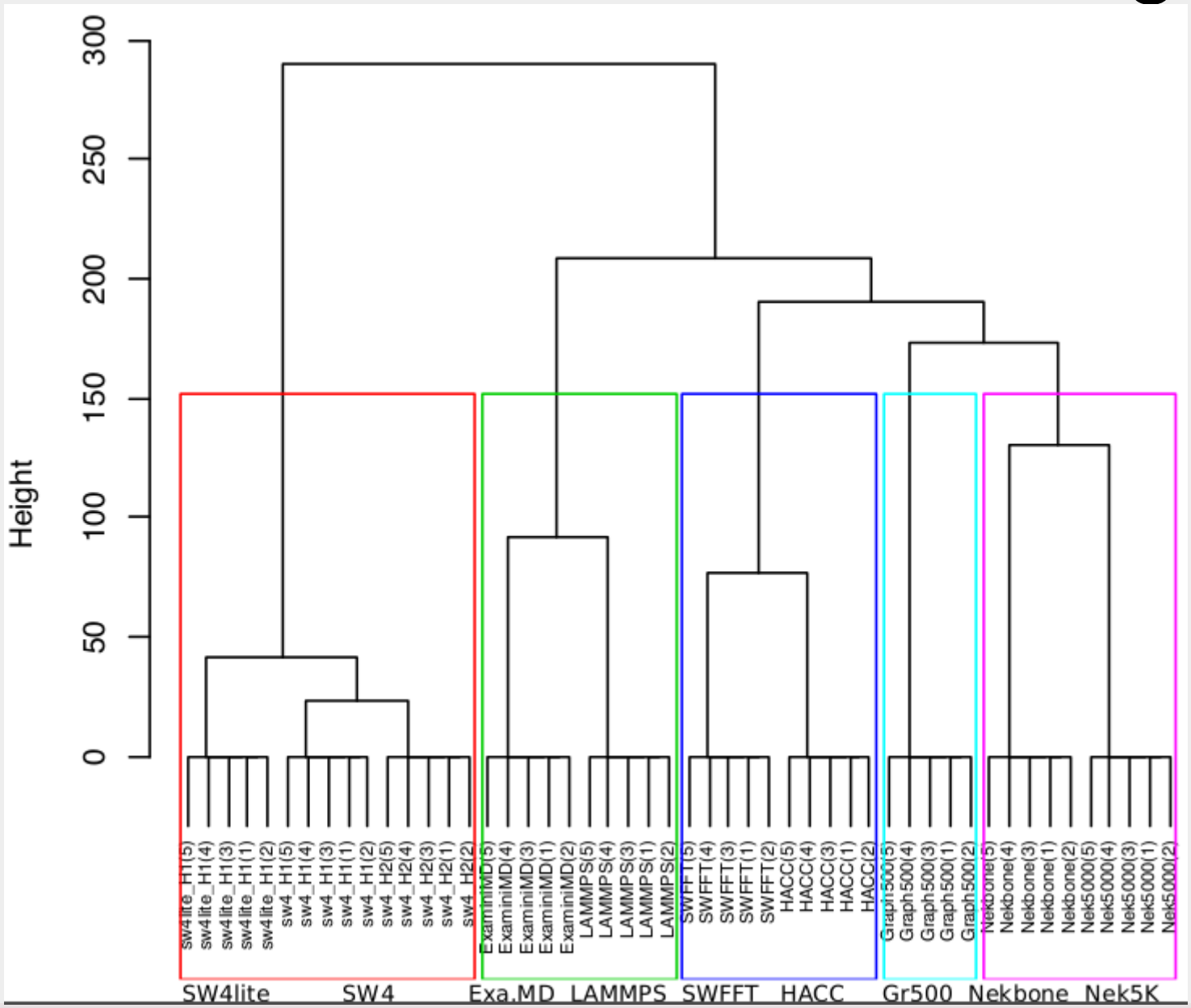




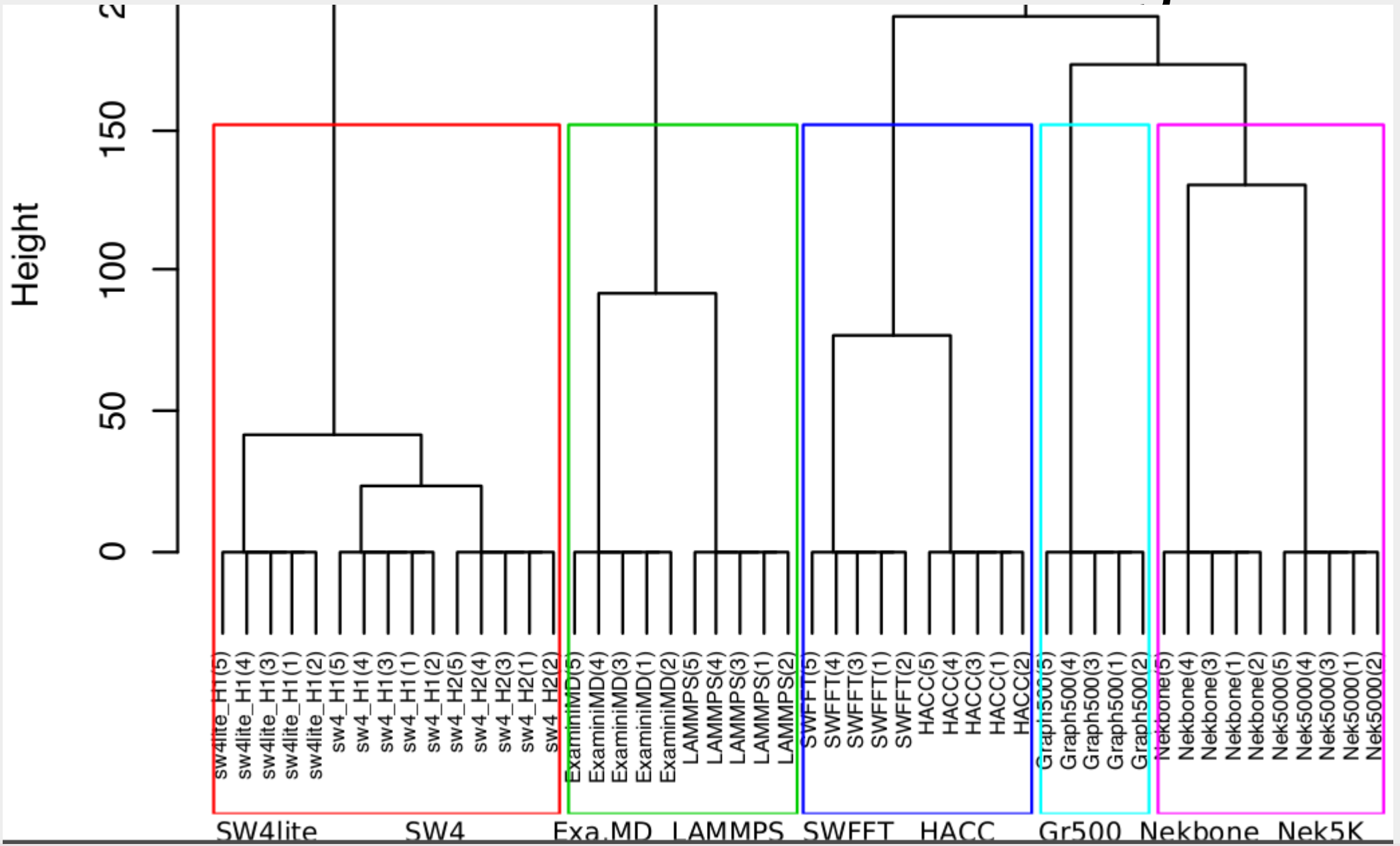
BW: Basic Node Clustering



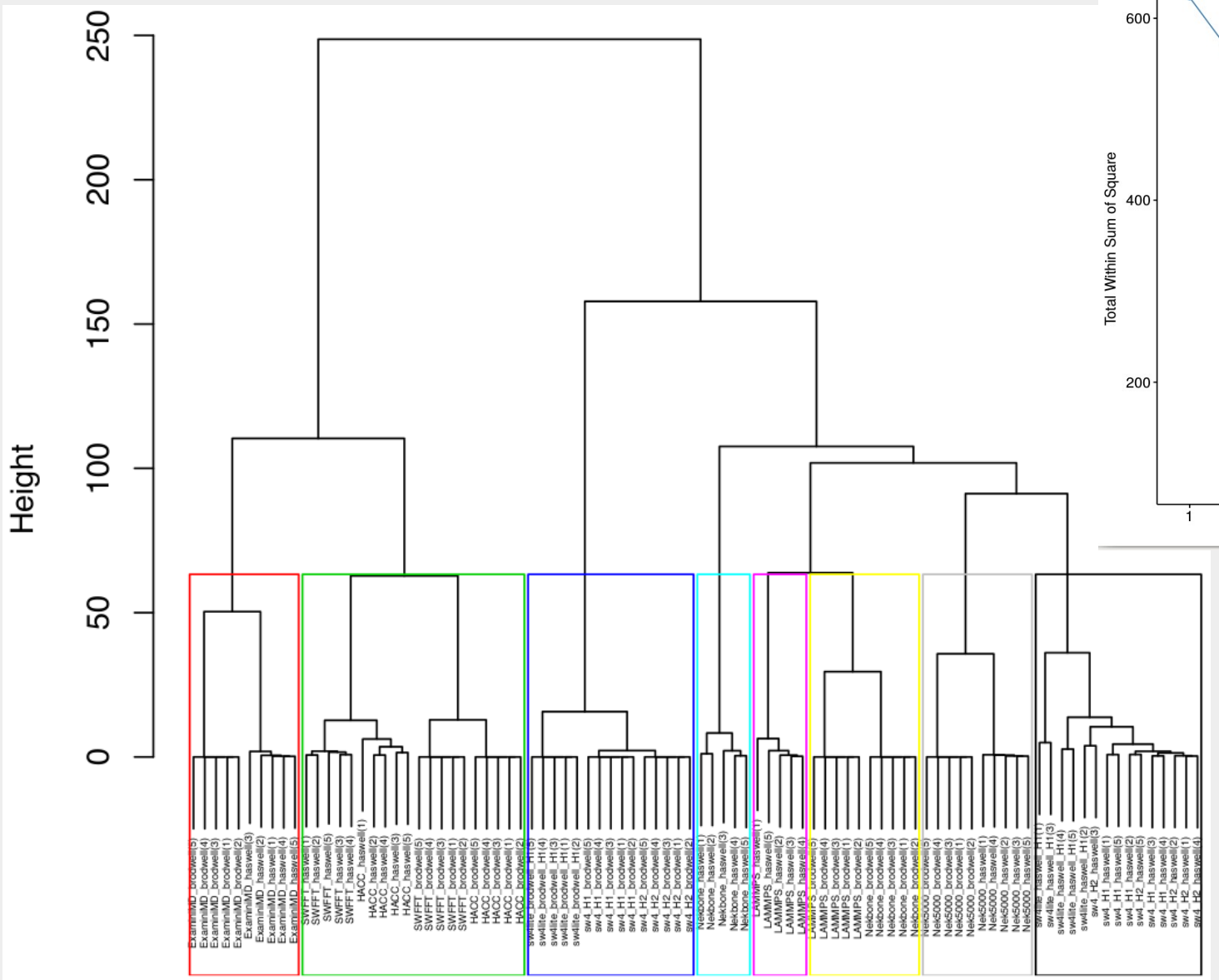
HW: Basic Node Clustering

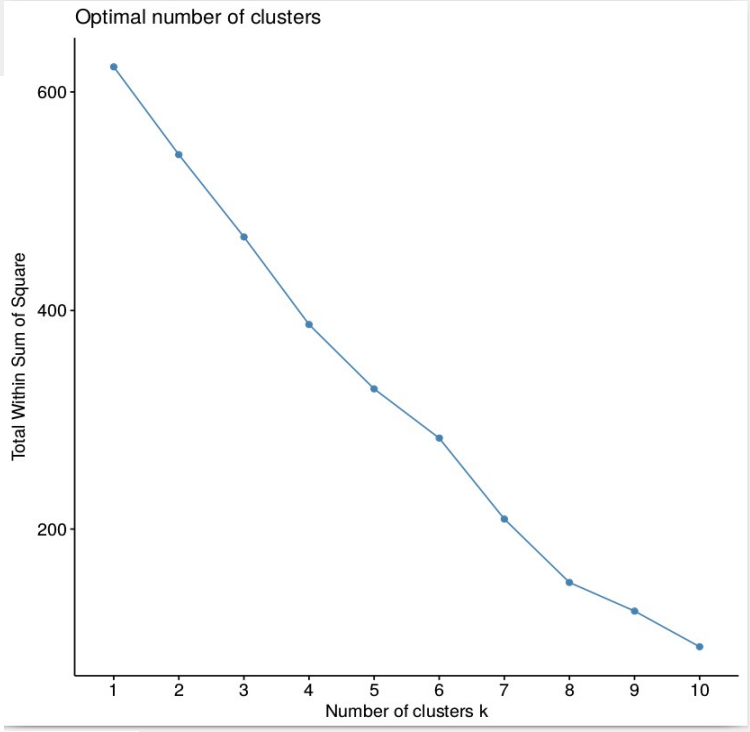


HW: Basic Node Clustering

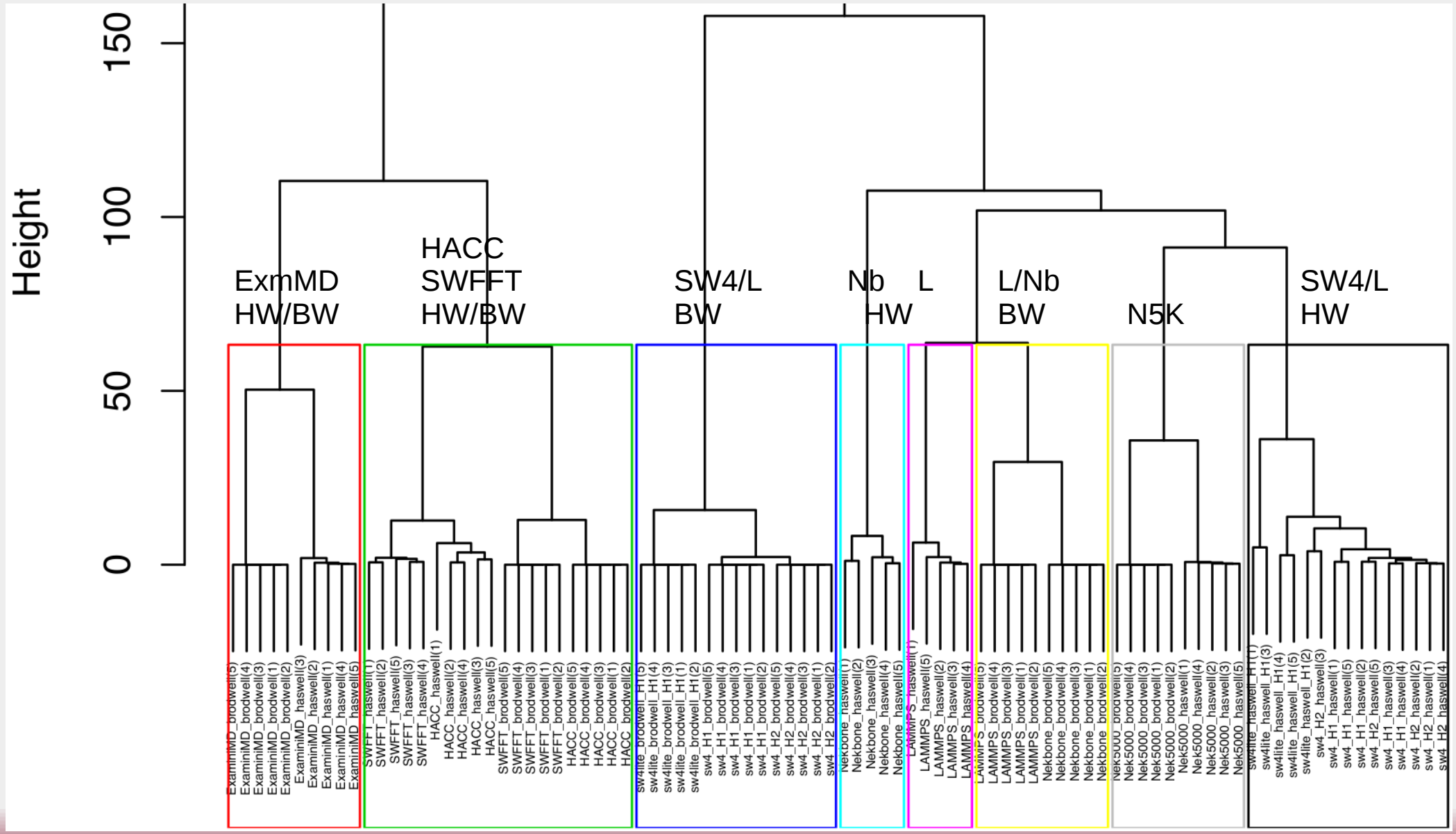


Both: Cluster





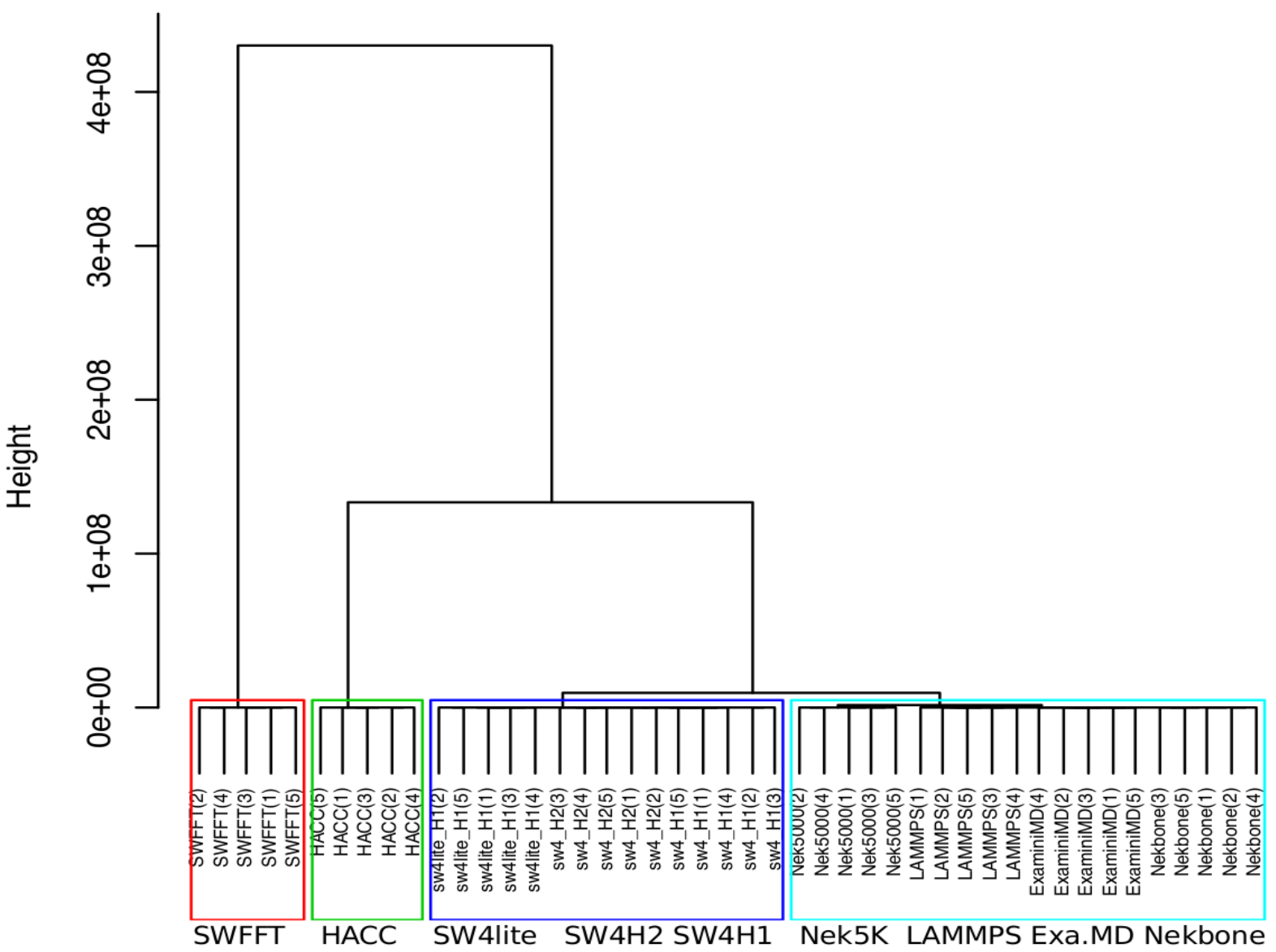
Both: Zoomed Clusters



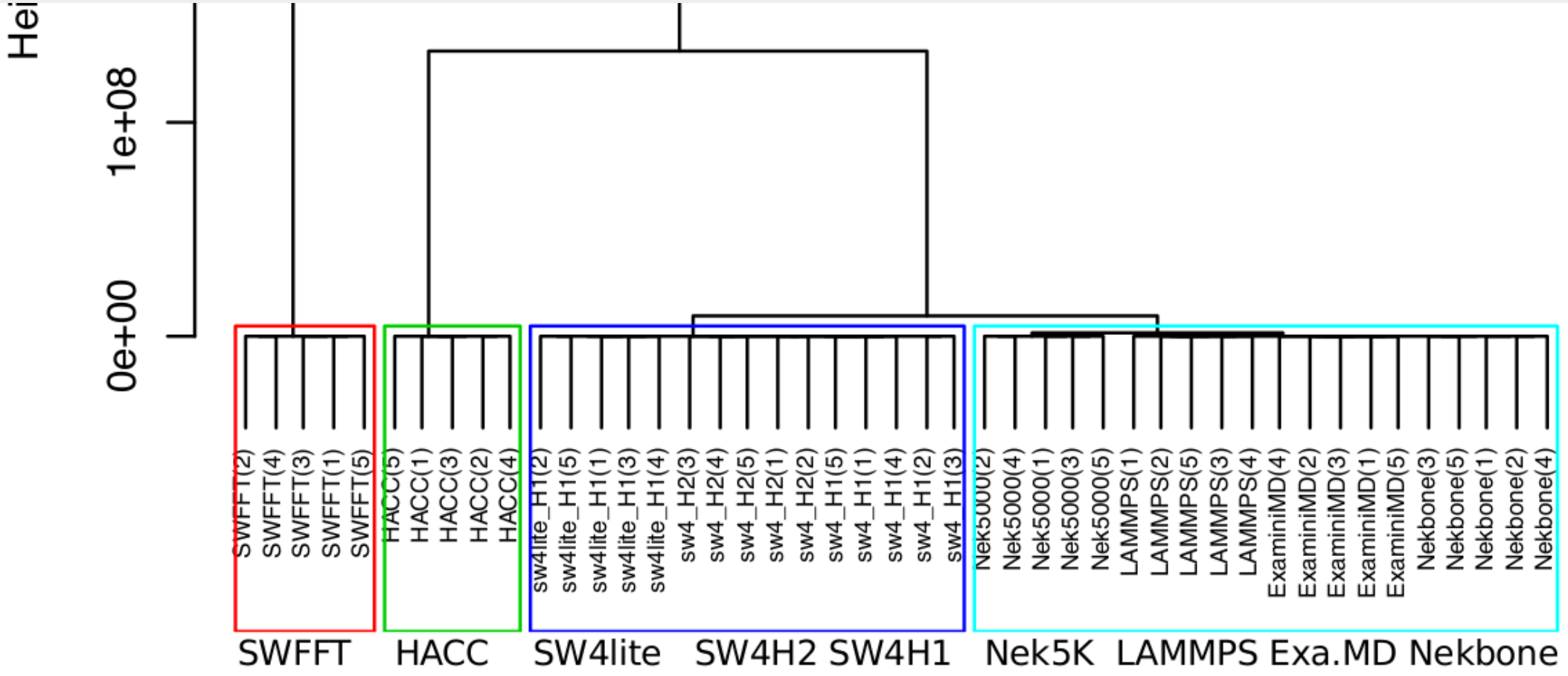
RD: Communication

- Used mpiP for data collection
  - % of time (app, mpi) for each MPI routine used
  - # of calls, # of bytes sent/received
- Four metrics
  - Apptime%, MPItime%, #calls/apptime, #bytes/apptime
- Four routine groups:
  - all_send, all_recv, all_multi, all_wait
- Data vector size: 14
  - all_wait * 2

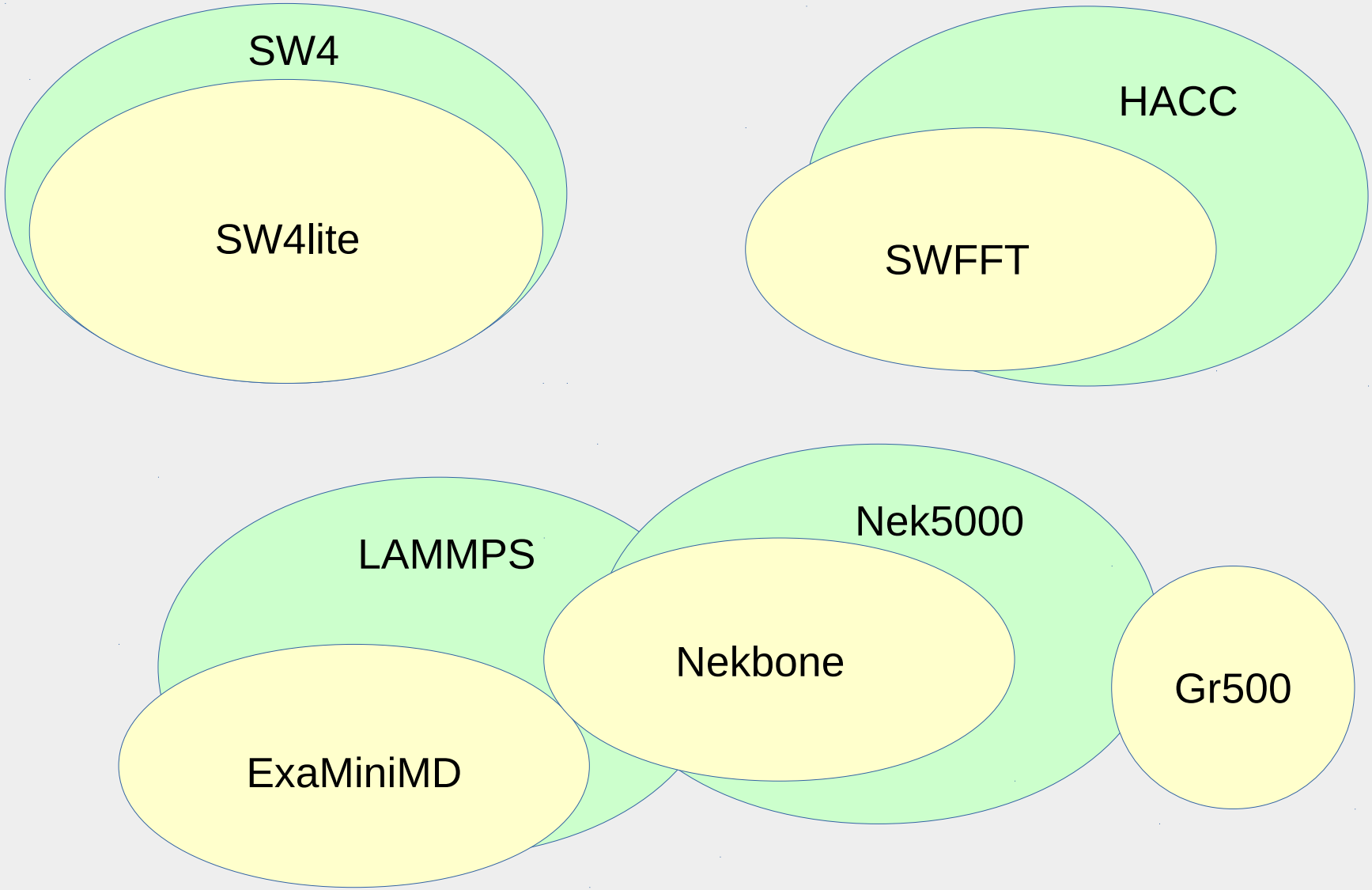
BW: Communication Clustering



BW: Communication Clustering



Do Proxies Match the Real Thing?

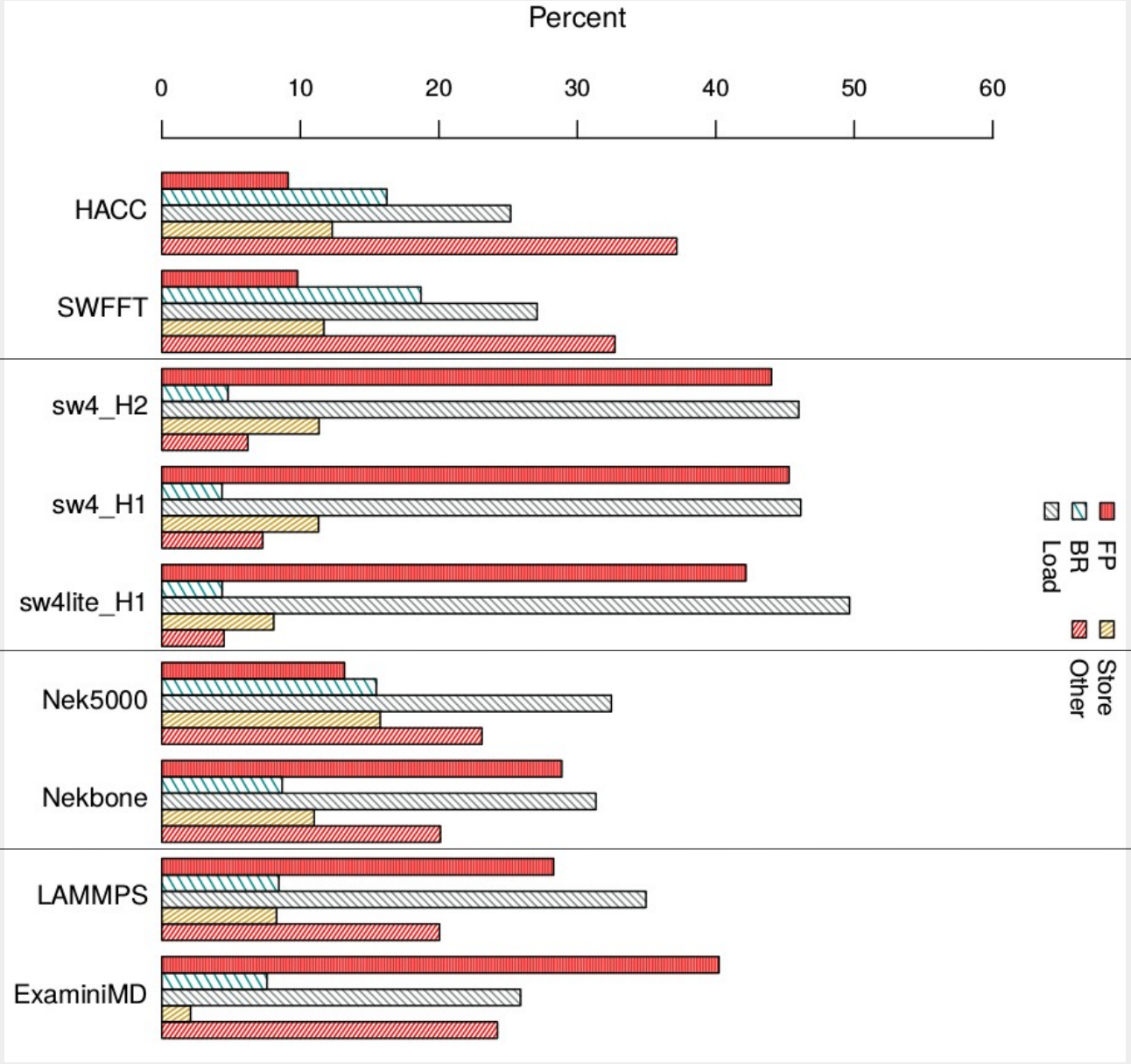


Questions?

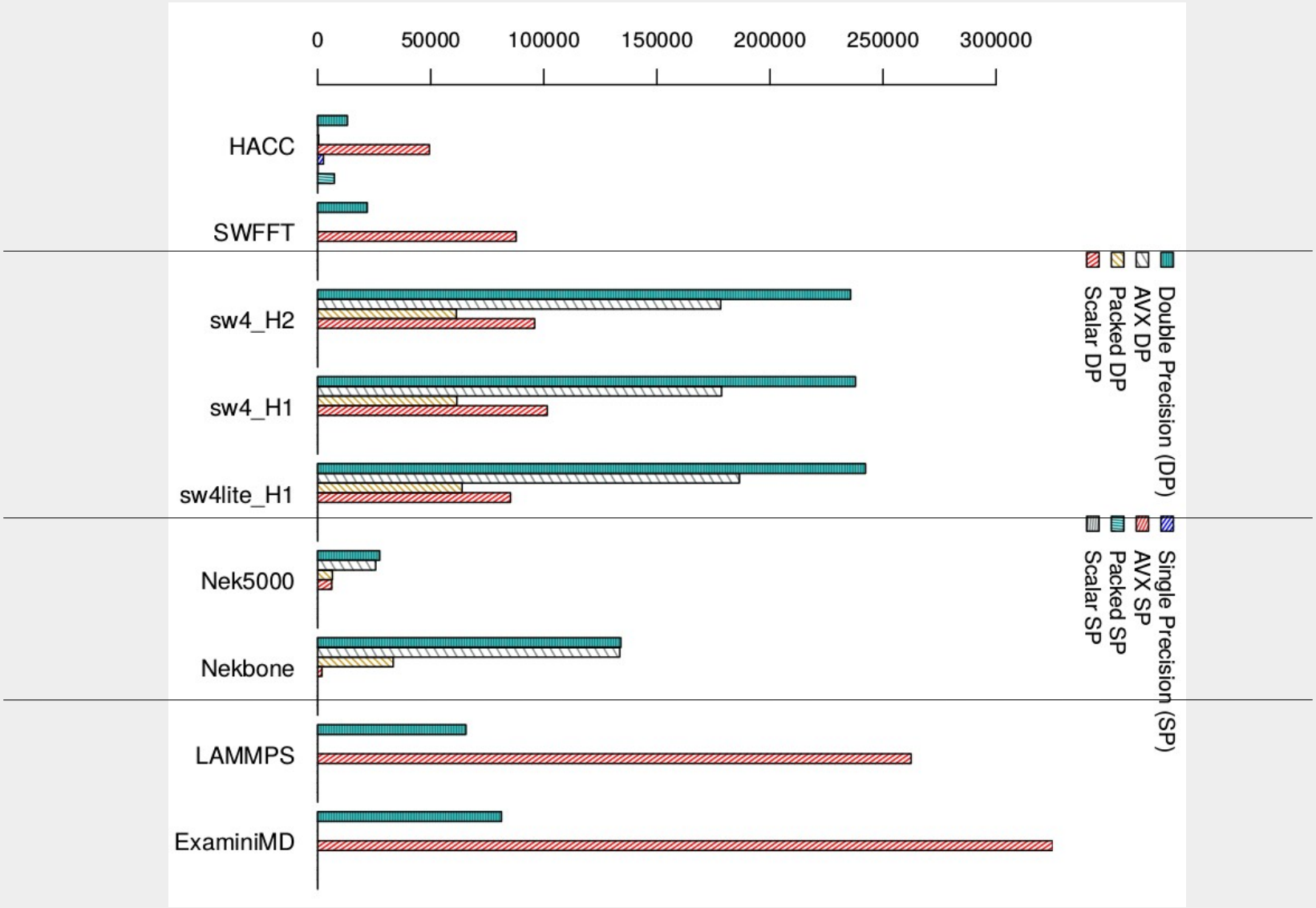
Future Work

- Further data analysis to identify points of difference
- Incorporate performance roofline models to identify where parent/proxy max out resource usage
- Match p/p communications to known patterns (e.g., seven comm. dwarves)
- If proxy is for limited piece of parent, limit parent data to that piece
- Incorporate other data for identifying and characterizing similar run configurations

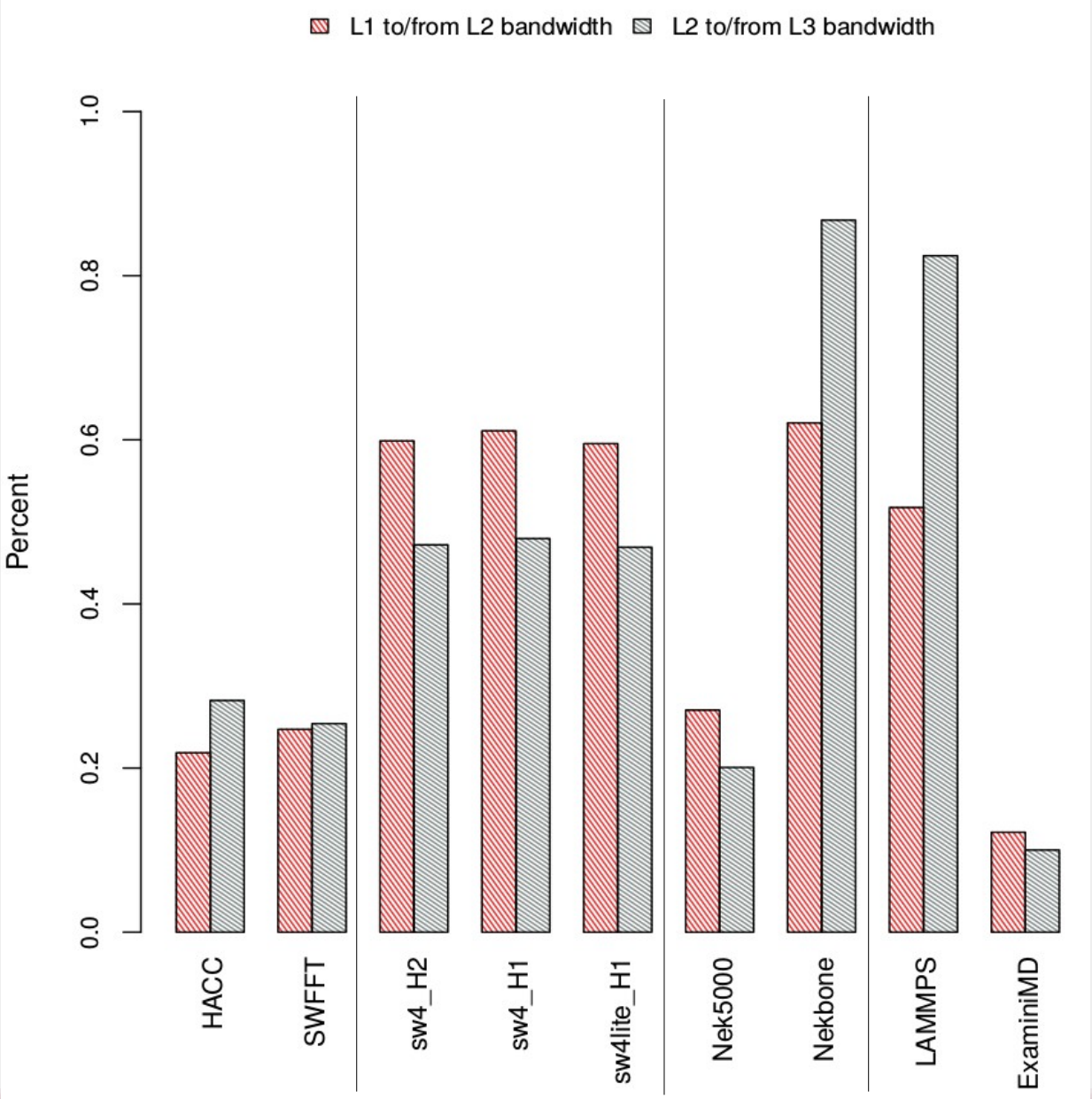
BW: Instruction Mix



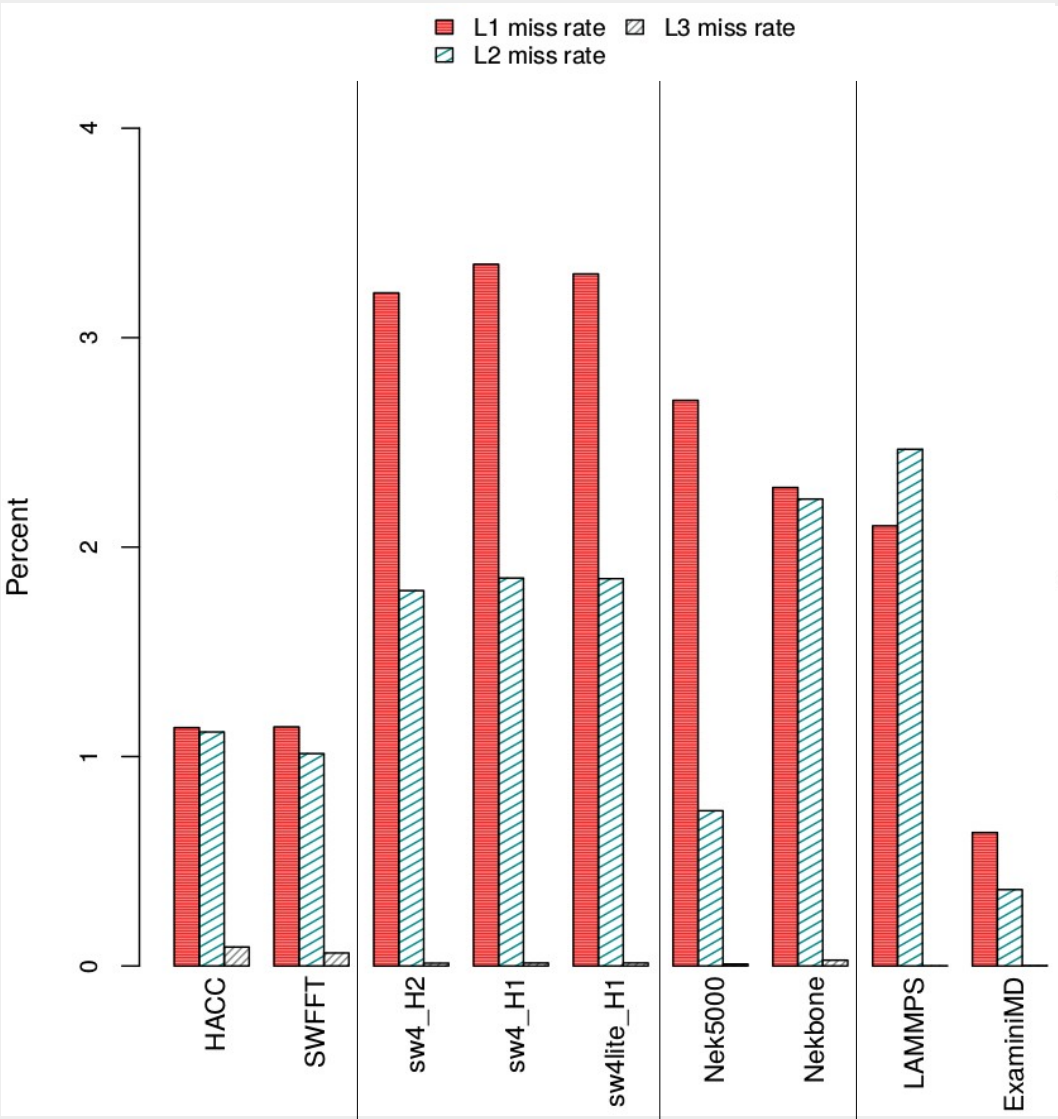
BW: FLOPS Mix

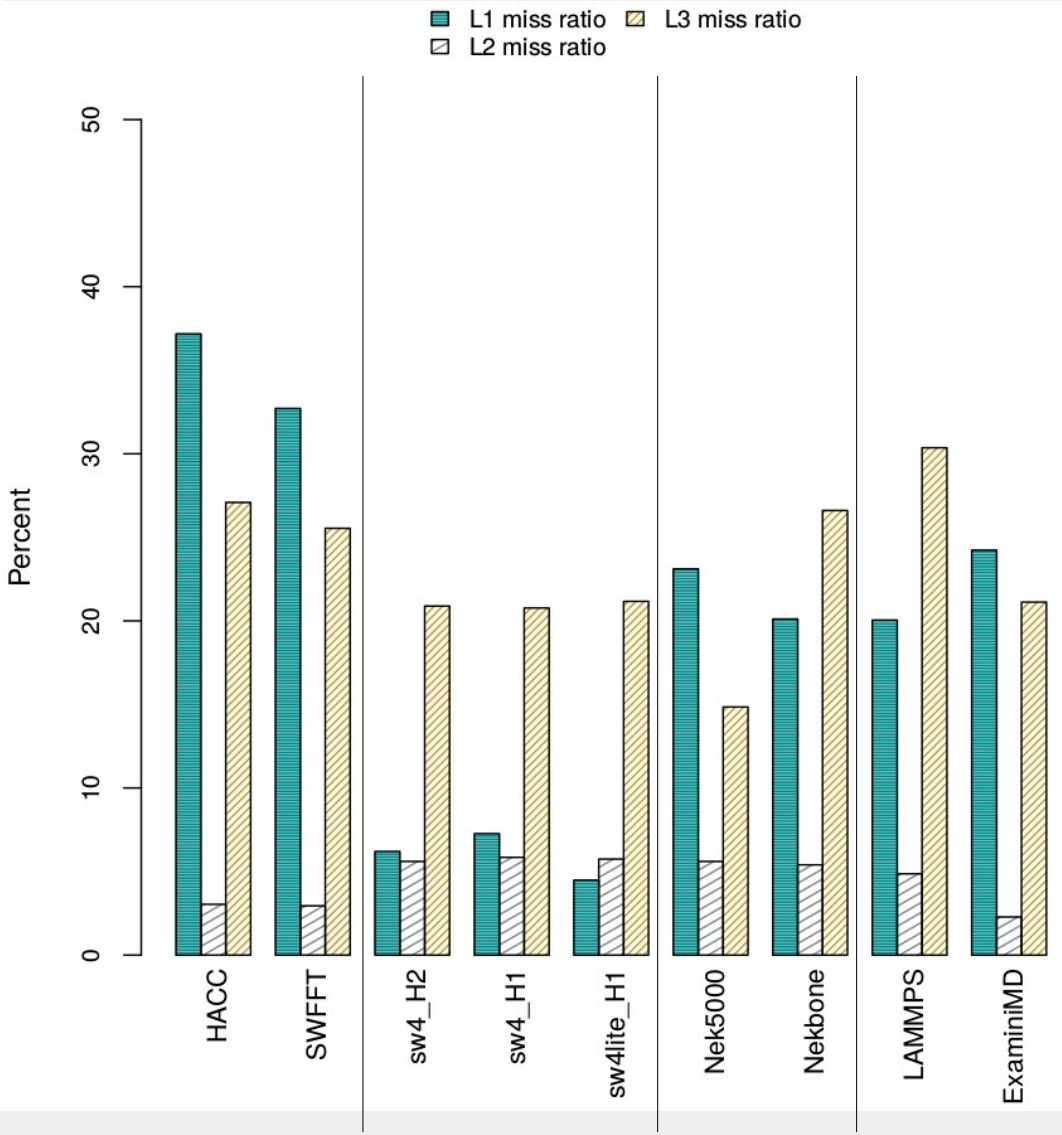

0
50000
100000
150000
200000
250000
300000
HACC
SWFFT
sw4_H2
sw4_H1
sw4lite_H1
Nek5000
Nekbone
LAMMPS
ExaminiMD
Double Precision (DP)
AVX DP
Packed DP
Scalar DP
Single Precision (SP)
AVX SP
Packed SP
Scalar SP

BW: L1-L2-L3 Bandwidth



BW: Miss Rates and Ratios

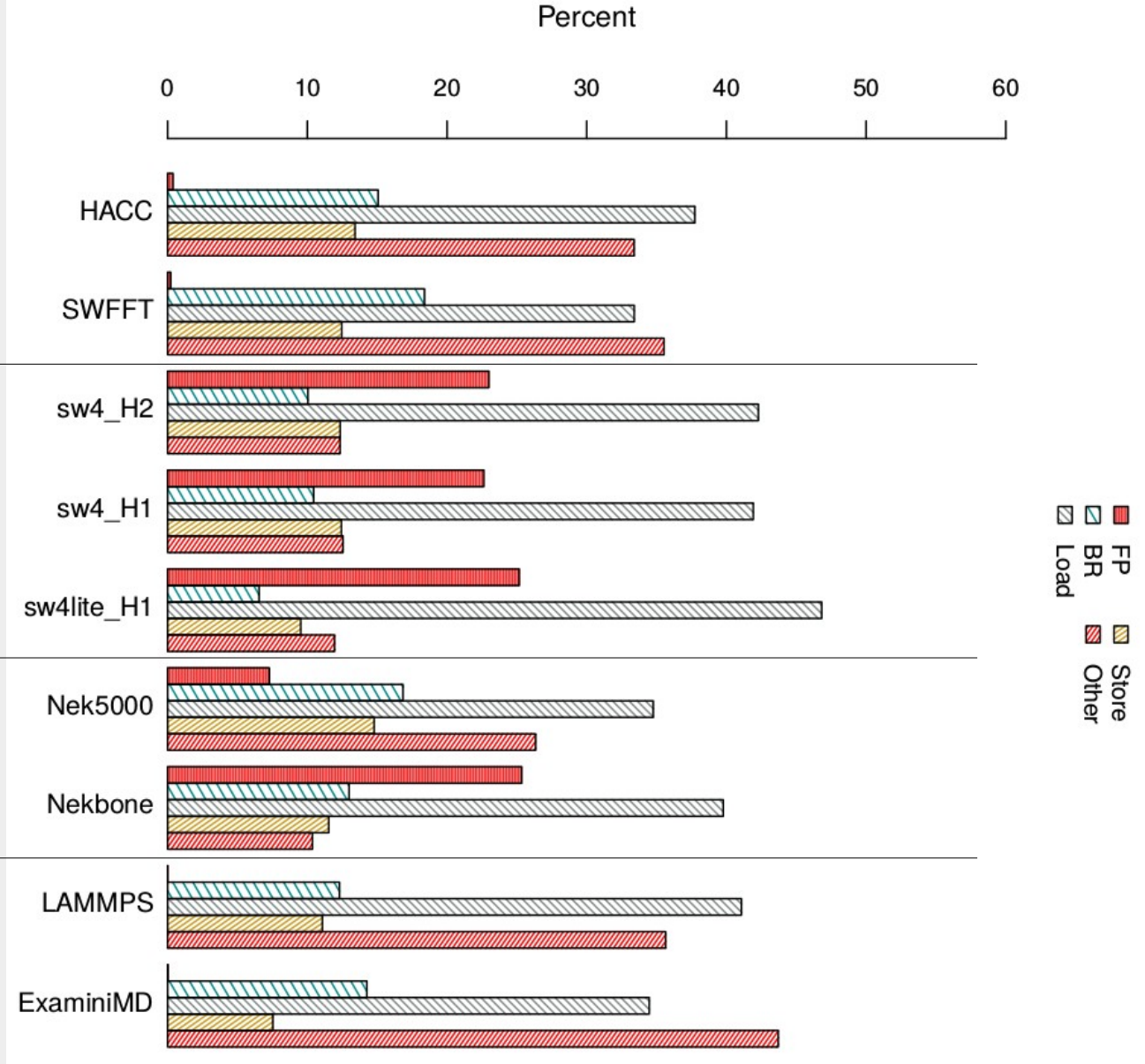




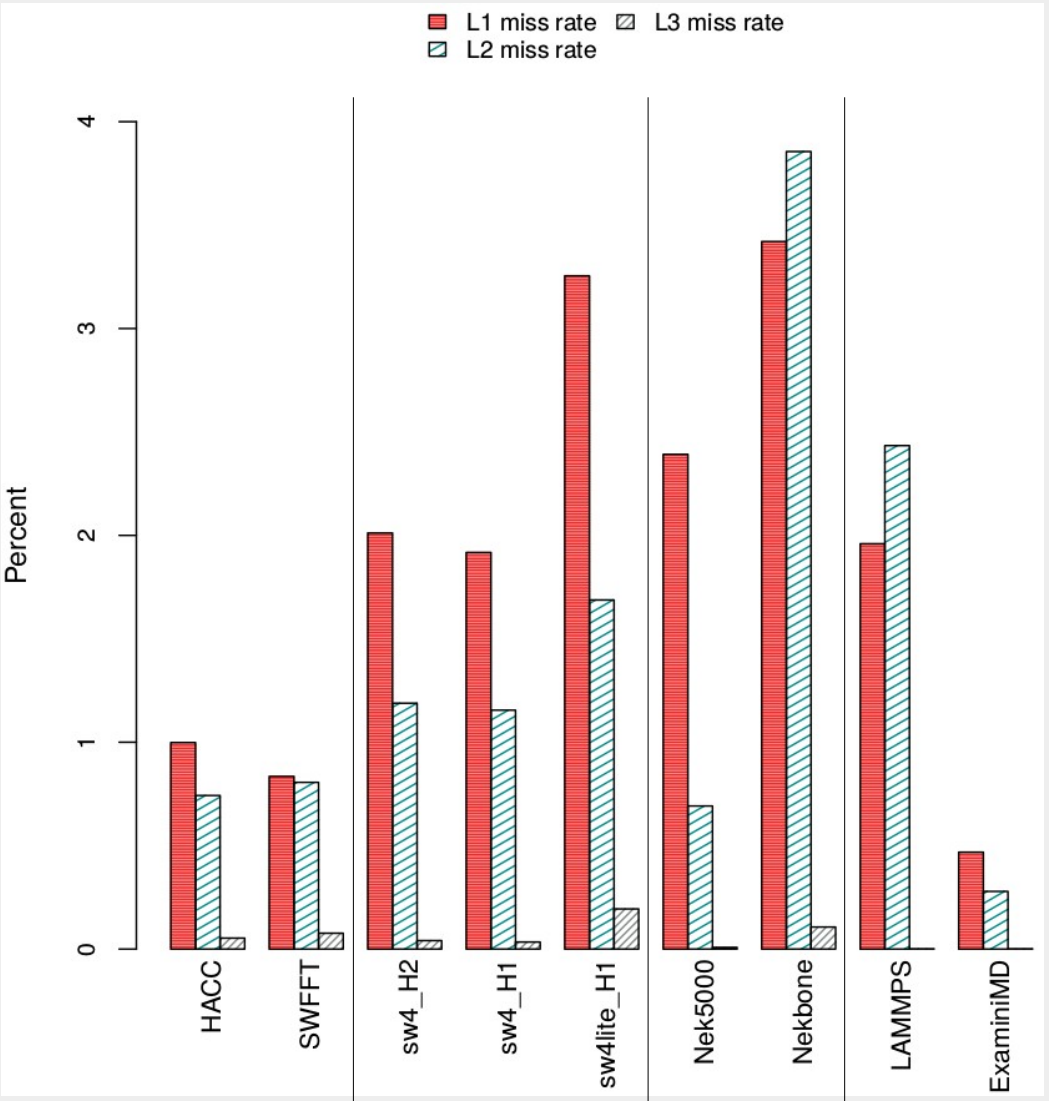
Basic Node Domain on Broadwell

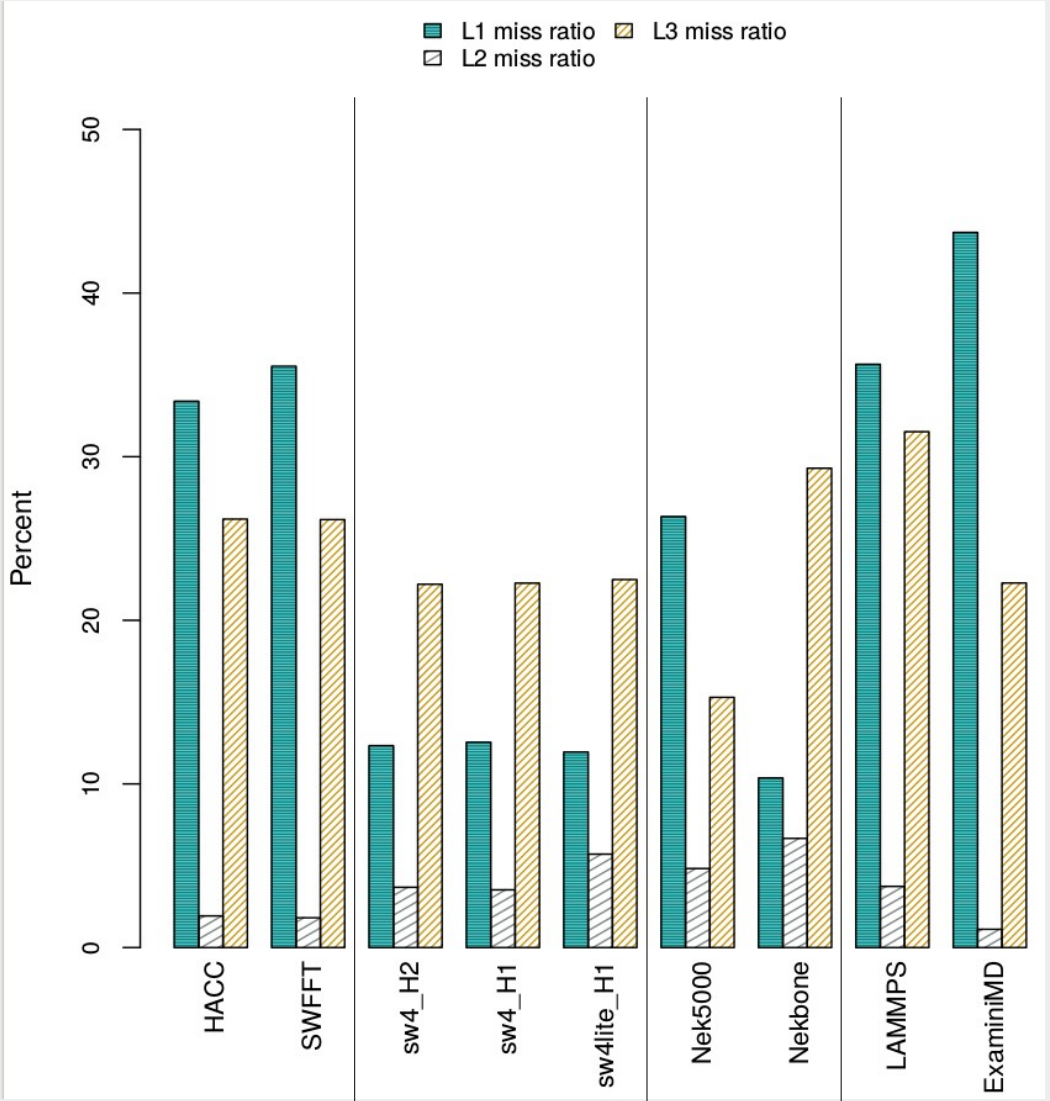
- SW4 and SW4lite are very similar
- HACC and SWFFT are very similar
- LAMMPS and ExaMiniMD are fairly similar
- Nek5000 and Nekbone are somewhat similar
  - cluster slighty after best-cluster fit
- Good: proxy always clusters first with parent
- Memory behavior is what is most divergent for N/N and L/X

HW: Instruction Mix

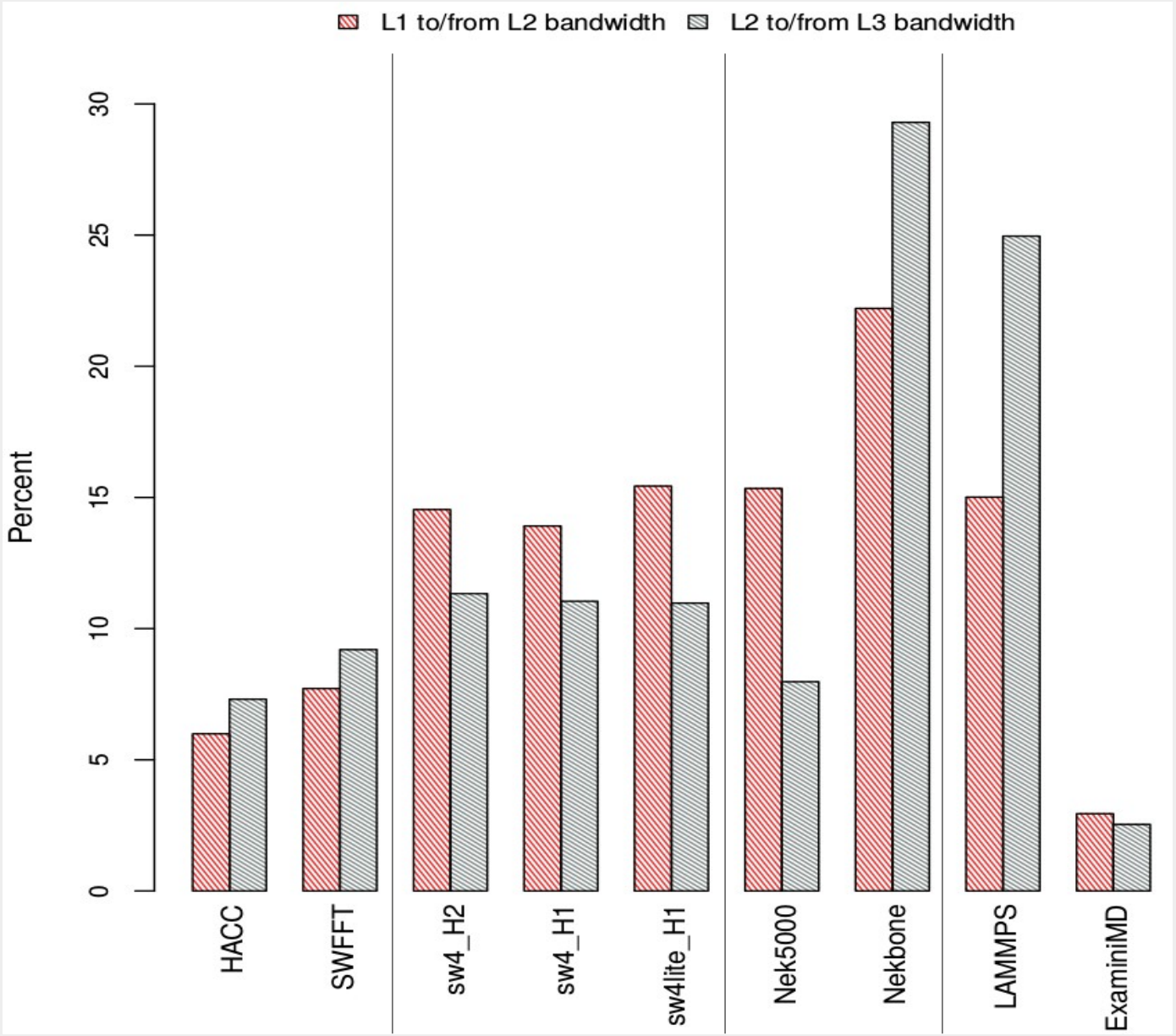


HW: Miss Rates & Ratios





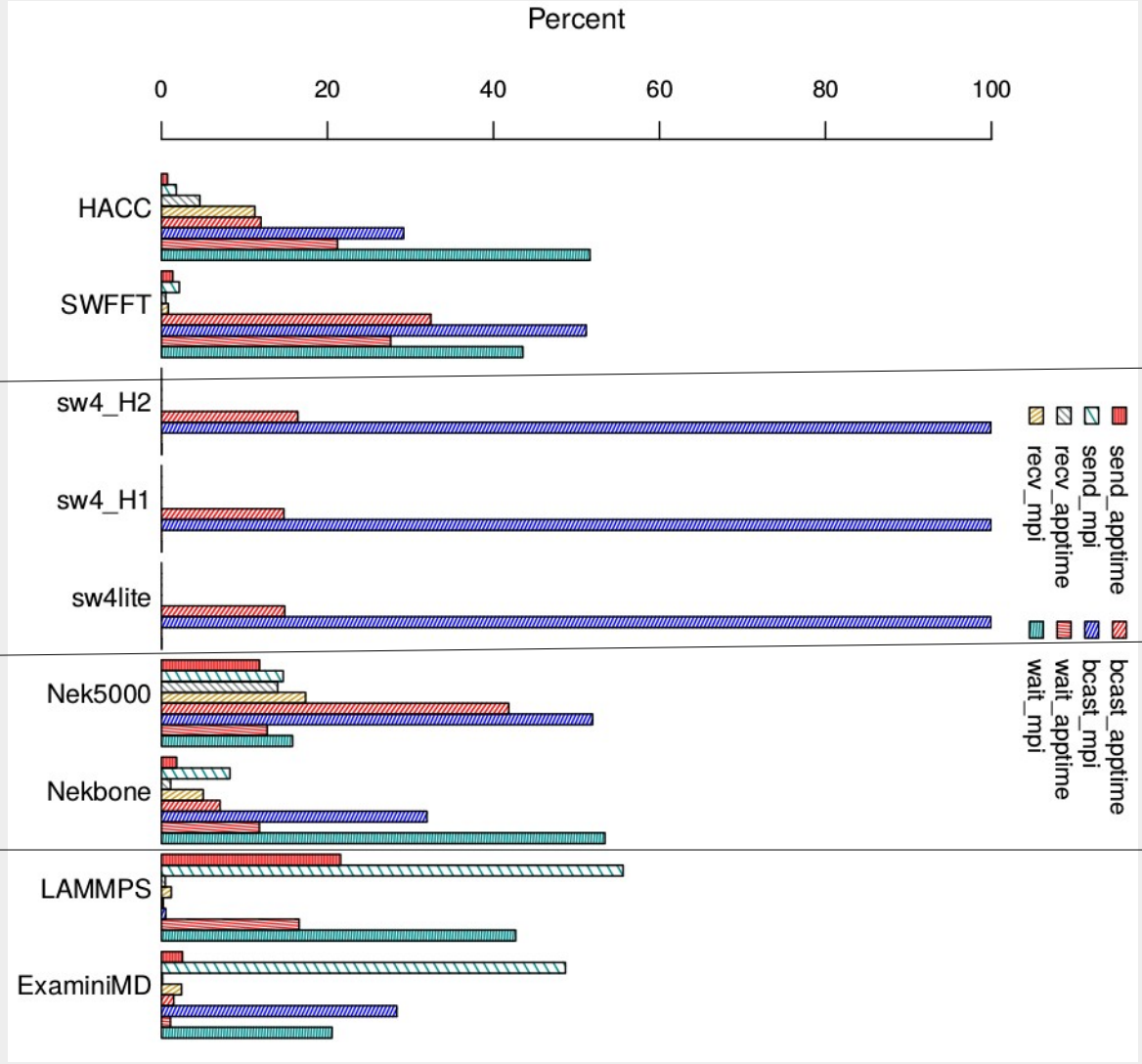
HW: Bandwidths



Basic Node on Haswell

- Good: same clustering order as on Broadwell
- Good: proxy always clusters first with parent
- Good: Graph500 clustered above all proxy/parent pair clusters
  - But not last
- Memory behavior again differentiates N/N and L/X

BW: MPI Times


Percent
0
20
40
60
80
100
HACC
SWFFT
sw4_H2
sw4_H1
sw4lite
Nek5000
Nekbone
LAMMPS
ExaminiMD
send_apptime
send_mpi
recv_apptime
recv_mpi
bcast_apptime
bcast_mpi
wait_apptime
wait_mpi

Communication Domain

- SW4 and SW4lite are very similar
- LAMMPS, ExaMiniMD, Nek500 and Nekbone are quite similar
  - Nekbone clusters with L/E before Nek5000
- HACC and SWFFT are very different from the rest, and from each other
- MpiP data vector not necessarily related to resource usage
  - And does not seem to be a good behavior separator