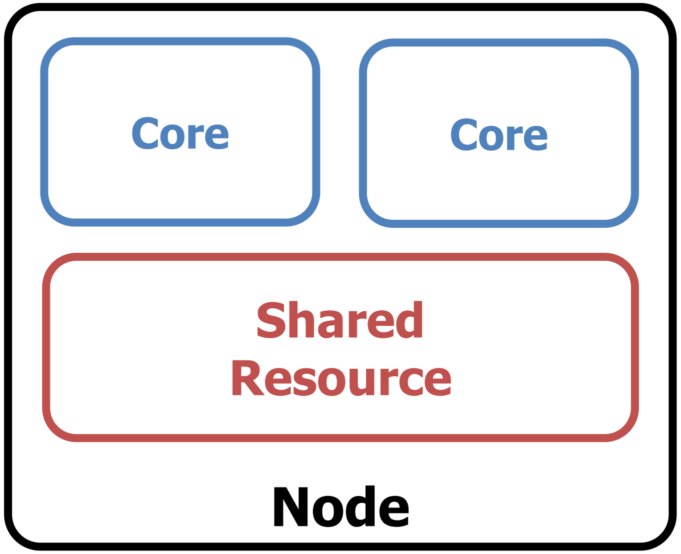
Interference Management for Distributed Parallel Applications in Consolidated Clusters

Jaeung Han[1], Seungheun Jeon[1], Young-ri Choi[2], and Jaehyuk Huh[1]

[1]School of Computing, KAIST
[2]School of Electrical and Computer Engineering, UNIST

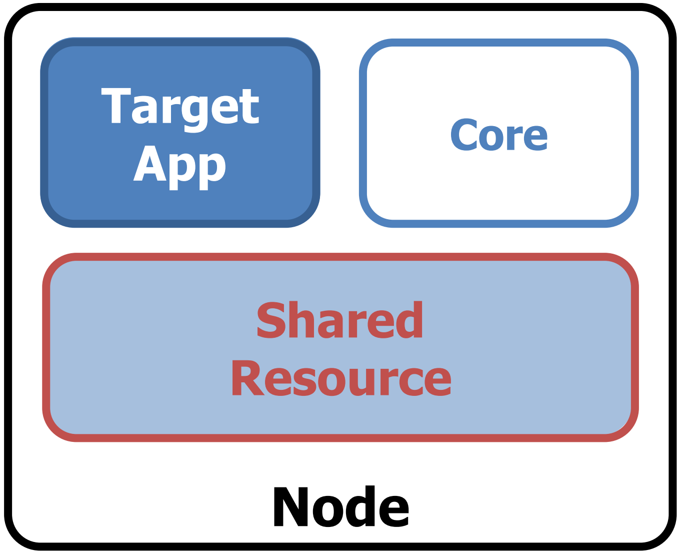
Interference in Consolidated Systems

- Resource contention causes performance interference
  - Last level caches, limited memory bandwidth, etc
- In single-node applications, the effect of intra-node interference is bounded within the system (node)



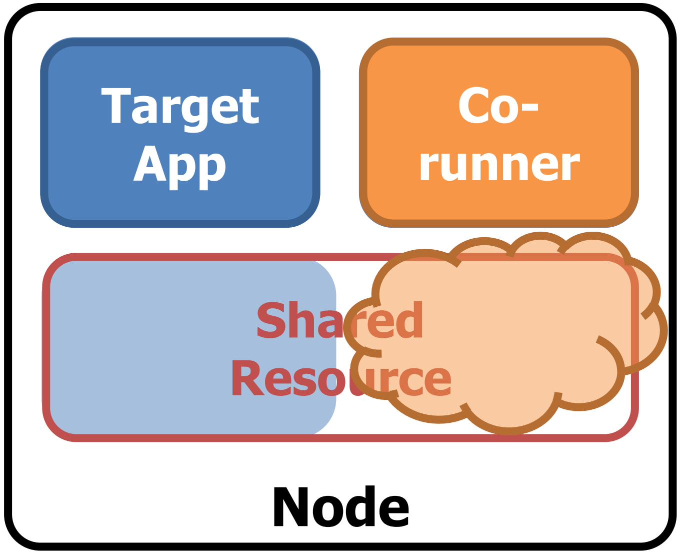
Interference in Consolidated Systems

- Resource contention causes performance interference
  - Last level caches, limited memory bandwidth, etc
- In single-node applications, the effect of intra-node interference is bounded within the system (node)



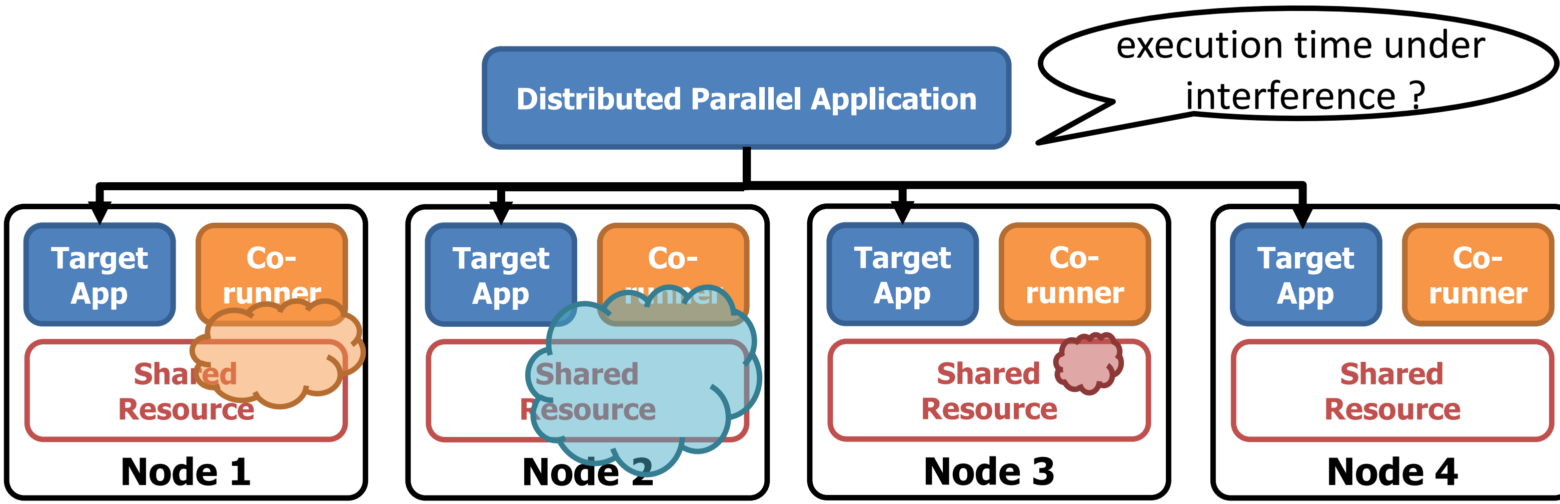
Interference in Consolidated Systems

- Resource contention causes performance interference
  - Last level caches, limited memory bandwidth, etc
- In single-node applications, the effect of intra-node interference is bounded within the system (node)



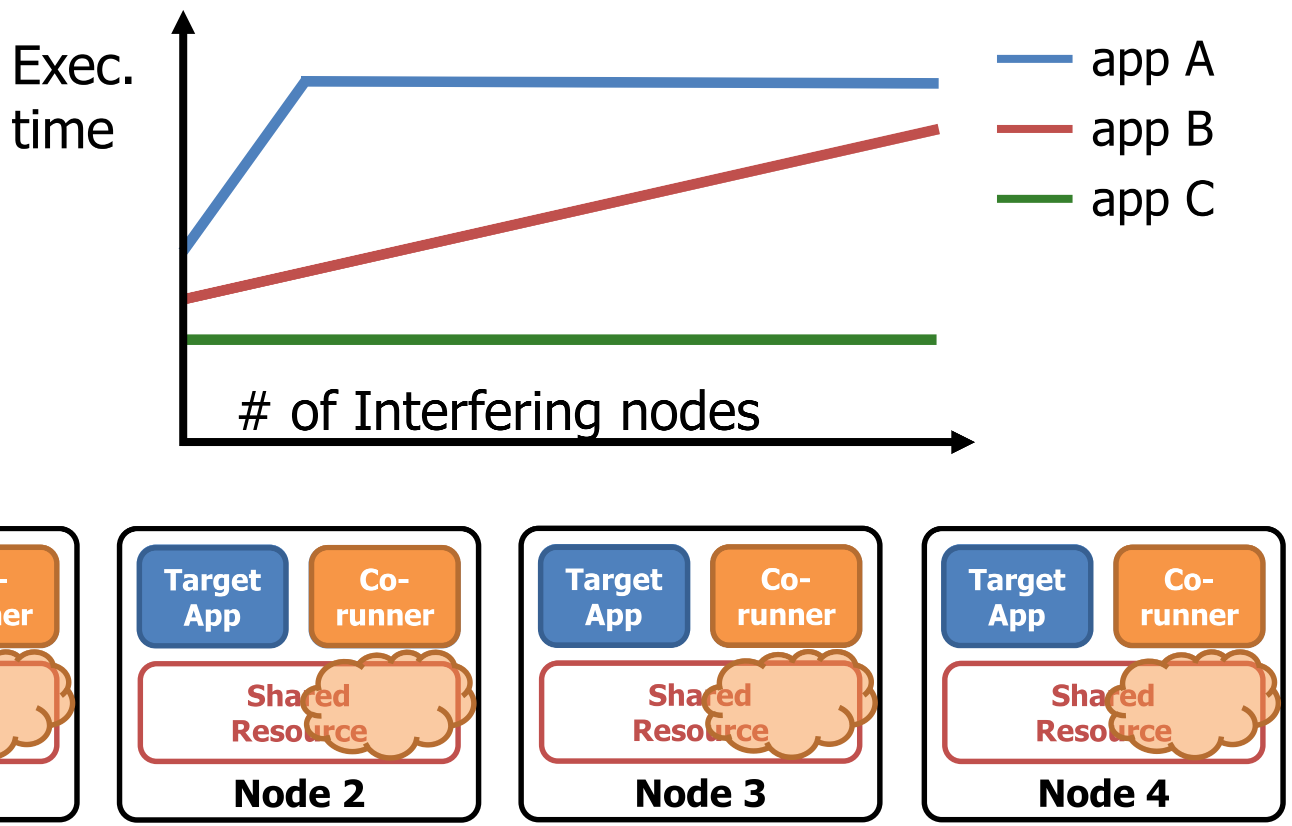
Interference in Consolidated Systems

- Resource contention causes performance interference
  - Last level caches, limited memory bandwidth, etc
- In single-node applications, the effect of intra-node interference is bounded within the system (node)
- In distributed applications, the interference effect from participating systems can interact with each other



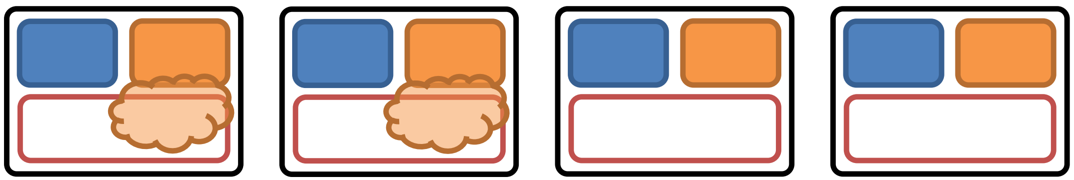
Interference in Distributed Applications

- Execution time increases by interference in participating nodes vary by application characteristics

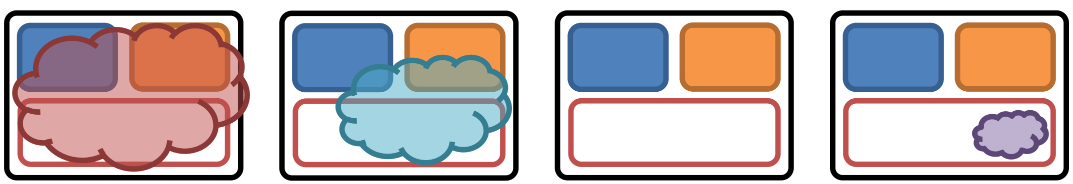


Challenges in Distributed Applications

- Can we estimate performance impact of interference for distributed parallel applications?
- *Two challenges*
  - Interference in a subset of nodes: interference propagation



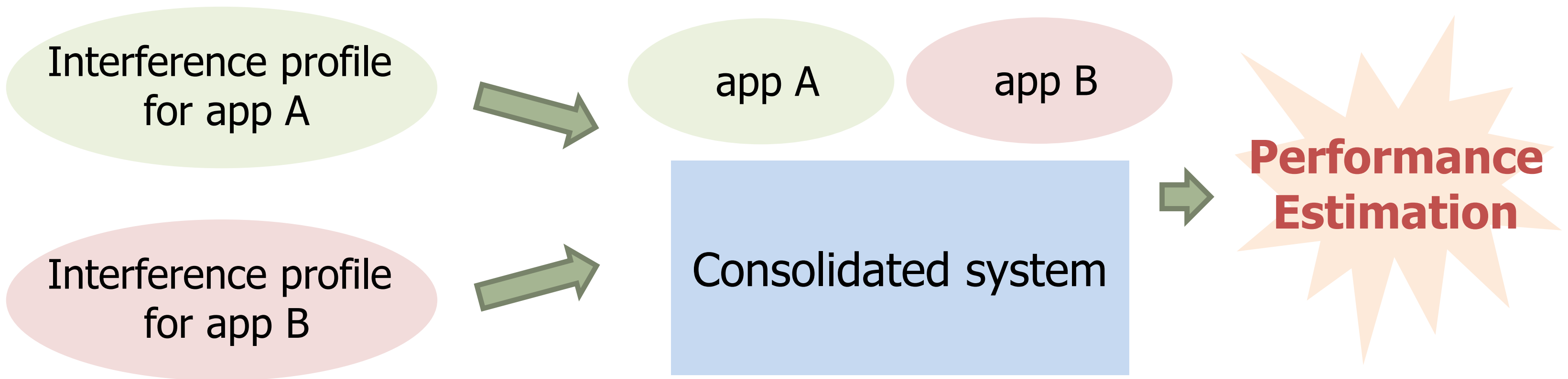
  - Different levels of interference: interference heterogeneity



- We propose a profiling-based interference estimation method

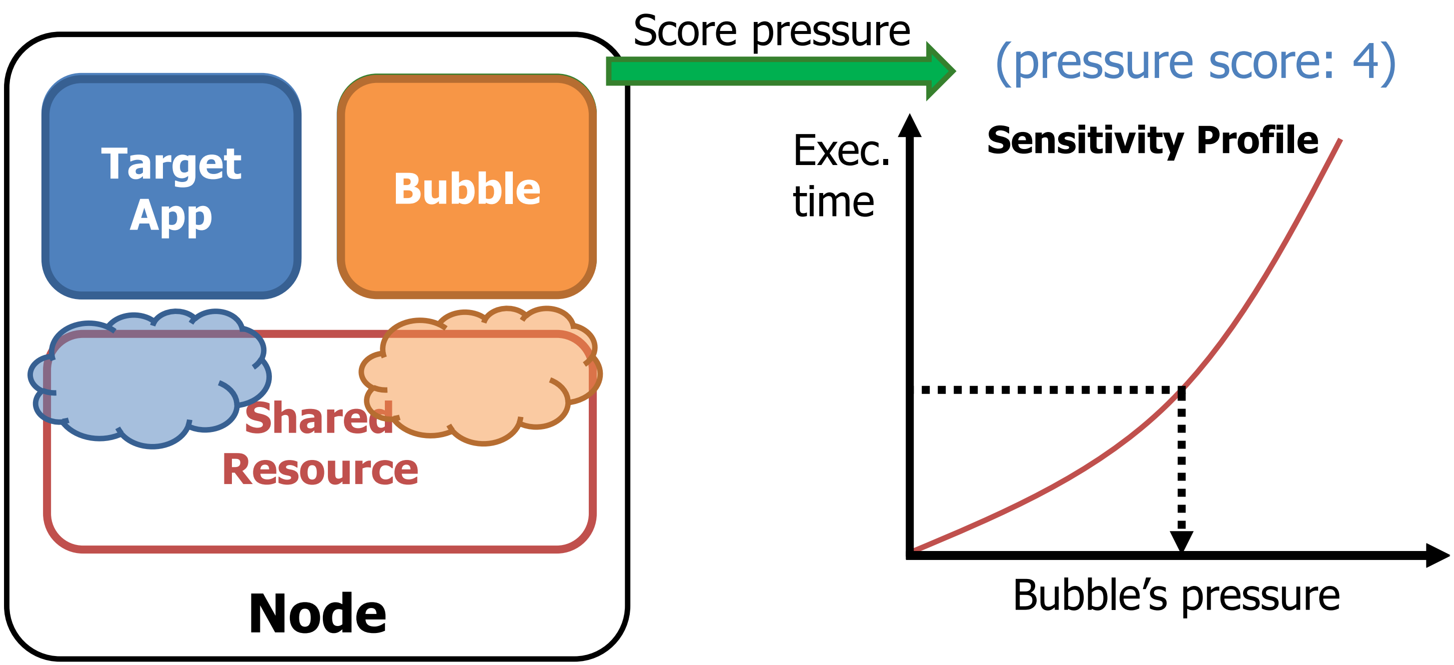
Quantifying Interference within a Node

- Bubble-Up [MICRO'11, Mars et al.]
  - Profiling-based interference model for single-node applications
  - Estimate the performance of co-located applications based on per-application interference profiles
- Per-application interference profile
  - Sensitivity profile: performance sensitivity to various levels of interference from the co-runner
  - Pressure score: interference level generated by the application



Bubble-Up Interference Profile

- Interference intensity is quantified to interference *pressure score*
- Bubble generates tunable amounts of interference pressure
- Reporter measures the pressure score (interference intensity generated by the application)



Bubble-Up Interference Profile

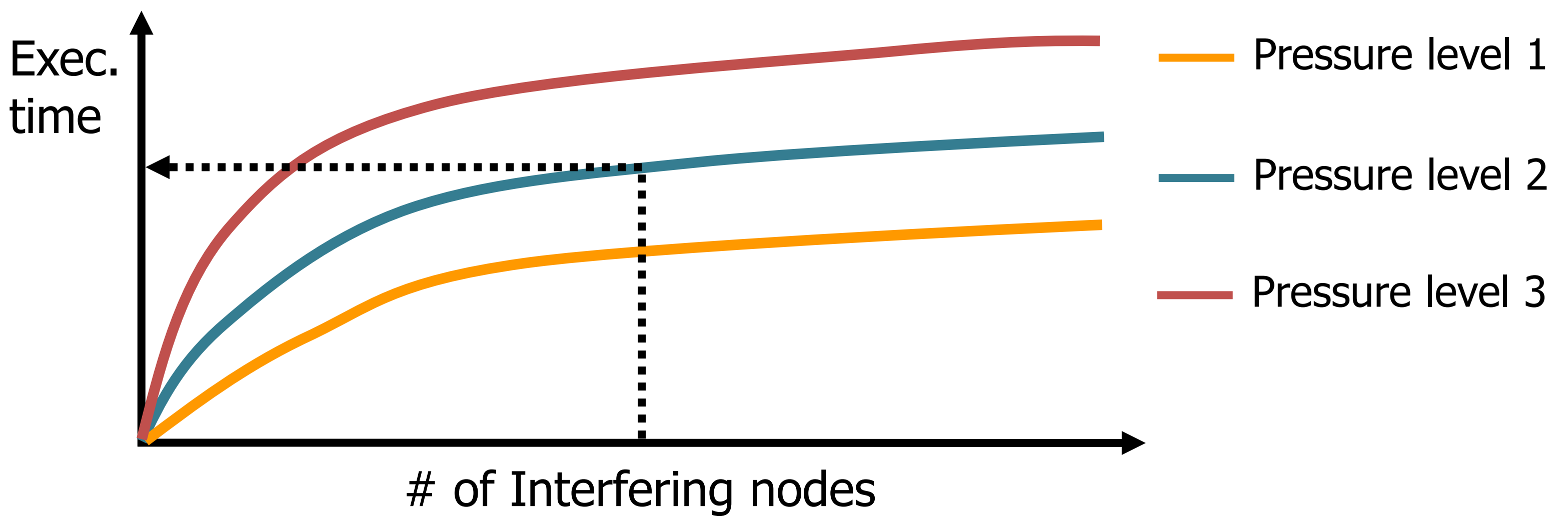
- Interference intensity is quantified to interference *pressure score*
- Bubble generates tunable amounts of interference pressure
- Reporter: measure the pressure score (interference intensity generated by the application)

Interference Profile for Distributed Applications

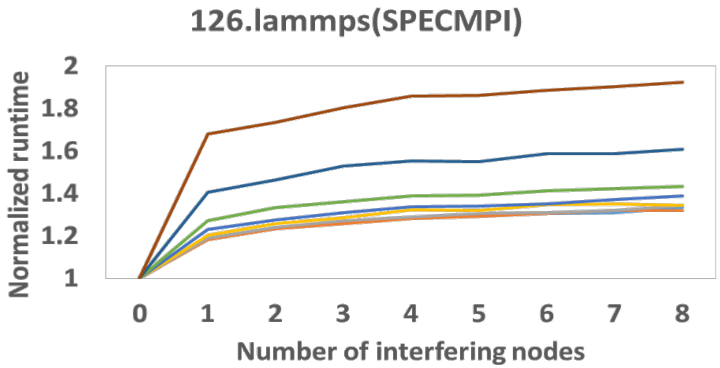
1) Pressure Score
2) Interference Propagation Profile
3) Heterogeneity Conversion Policy

Propagation in Distributed Applications

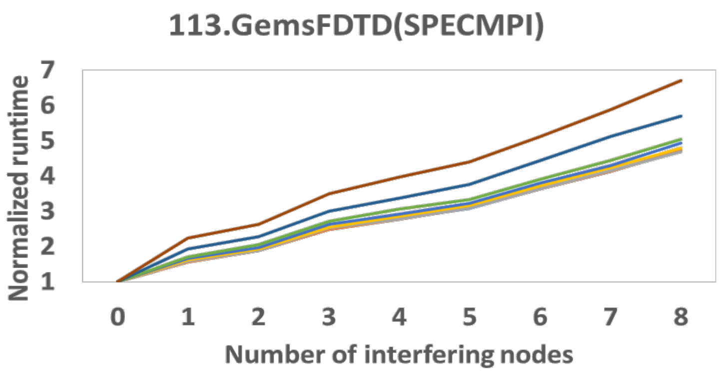
- Interference on a subset of nodes can slow down the execution progress in non-interfering nodes
- Interference propagation profile
  - Execution time changes by the number of interfering nodes
  - Each node suffers from the same level of interference



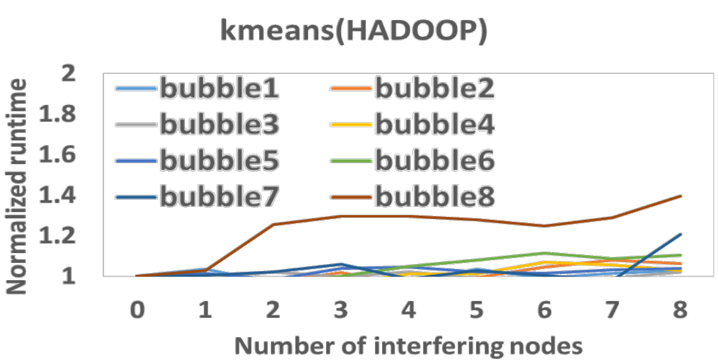
Common Interference Propagation Patterns



- High propagation
  - One interfering node affects the exec. time significantly
  - 104.milc, 126.lammps ...



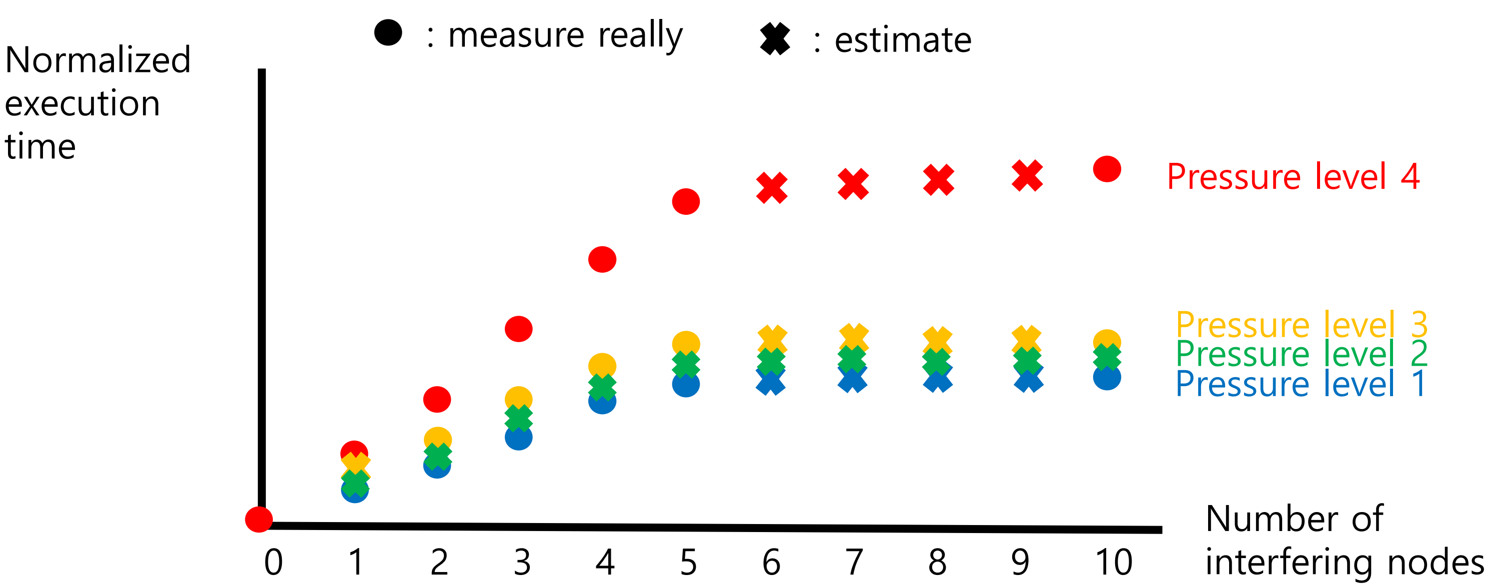
- Proportional propagation
  - Exec. time increases proportionally
  - 113.GemsFDTD ...



- Low propagation
  - Resilient to the interference
  - Kmeans(HADOOP), PageRank(SPARK) ...

Reducing Profiling Runs

- Binary-optimized
  - Shapes of curves are similar, regardless of pressure levels
  - Interpolate the exec. time from # of interfering nodes and pressure levels



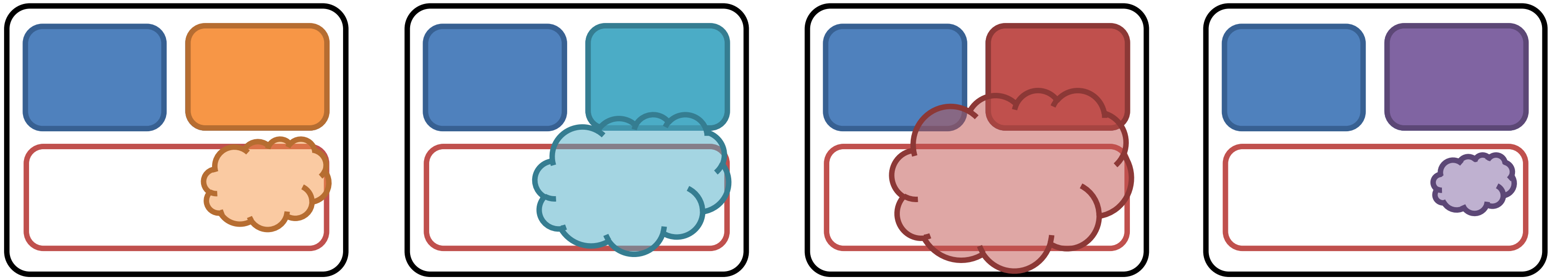
Interference Propagation

- Binary-optimized
  - Shape of curves are similar, regardless of bubble pressures
  - Extrapolate the exec. time from # interfering nodes and bubble pressures

Binary-optimized only need 18.45% of total profiling space with 3.16% error

Interference Heterogeneity

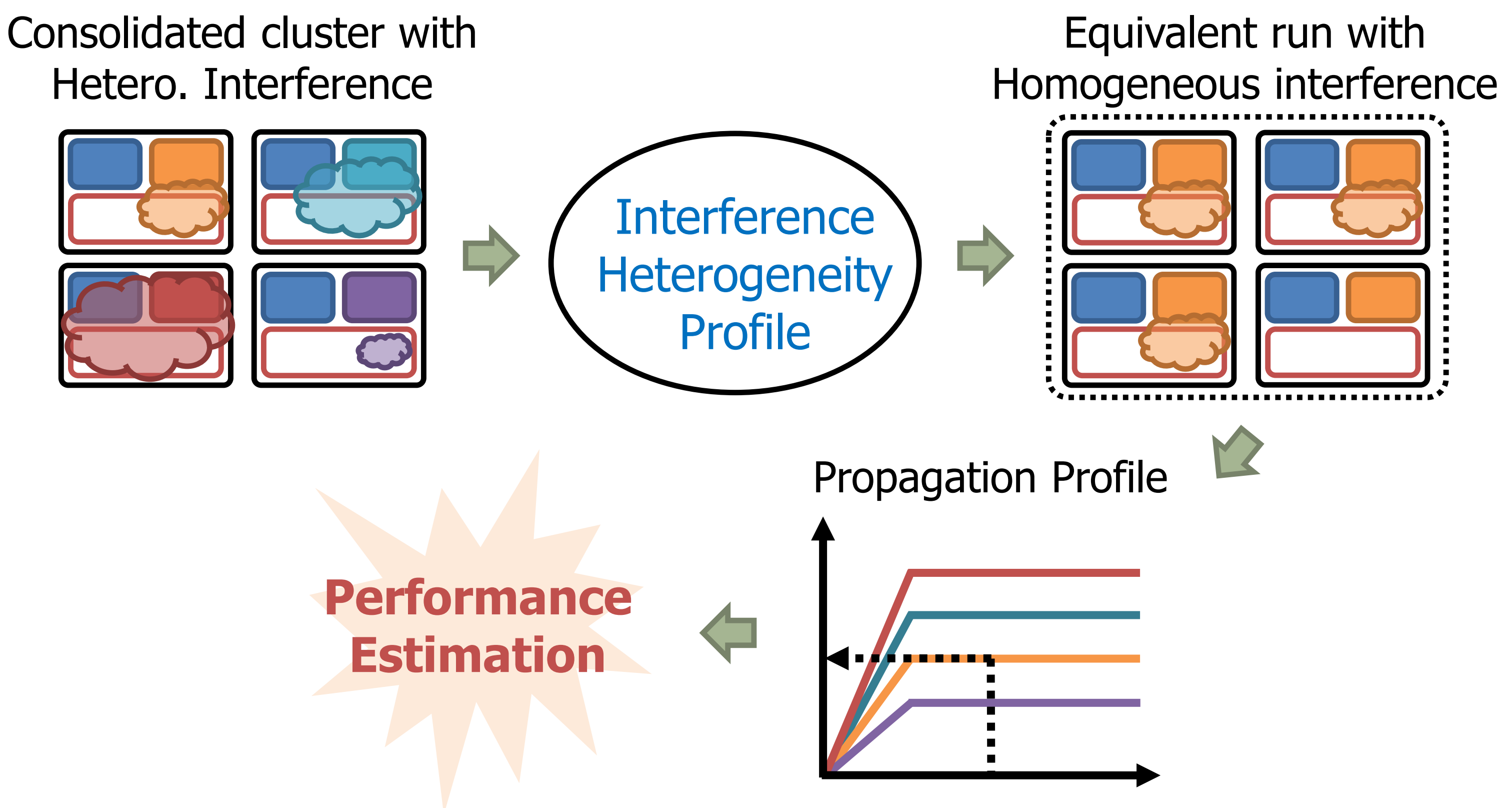
- Each node can suffer with different interference intensity



- Too large space for profiling all possible heterogeneous interferences
  - 4 nodes + 9 interference levels : 495
  - 8 nodes + 9 interference levels : 12,870
  - 32 nodes + 9 interference levels : 76,904,685

Interference Heterogeneity Profile

- Interference Heterogeneity Profile
  - Convert heterogeneous interference to an equivalent hypothetical run with homogeneous interference



Conversion Policies

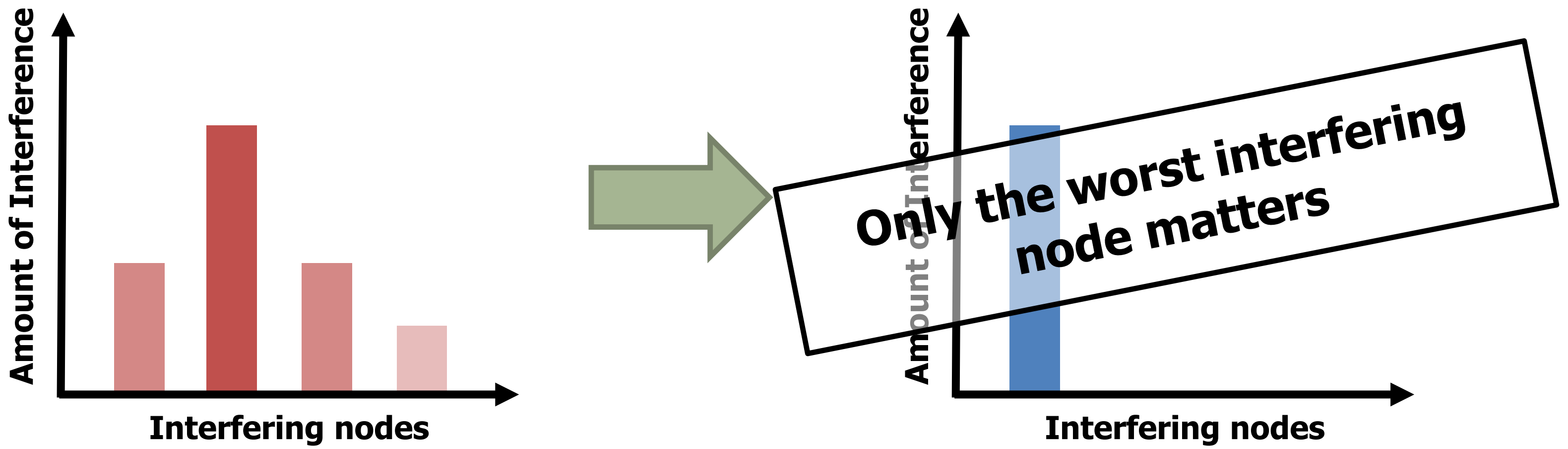
- 4 available conversion policies
  - N max
  - N+1 max
  - All max
  - Interpolate

- Evaluate all policies during profiling runs, and pick the best one for each application

- Use random sampling to reduce the number of profiling runs

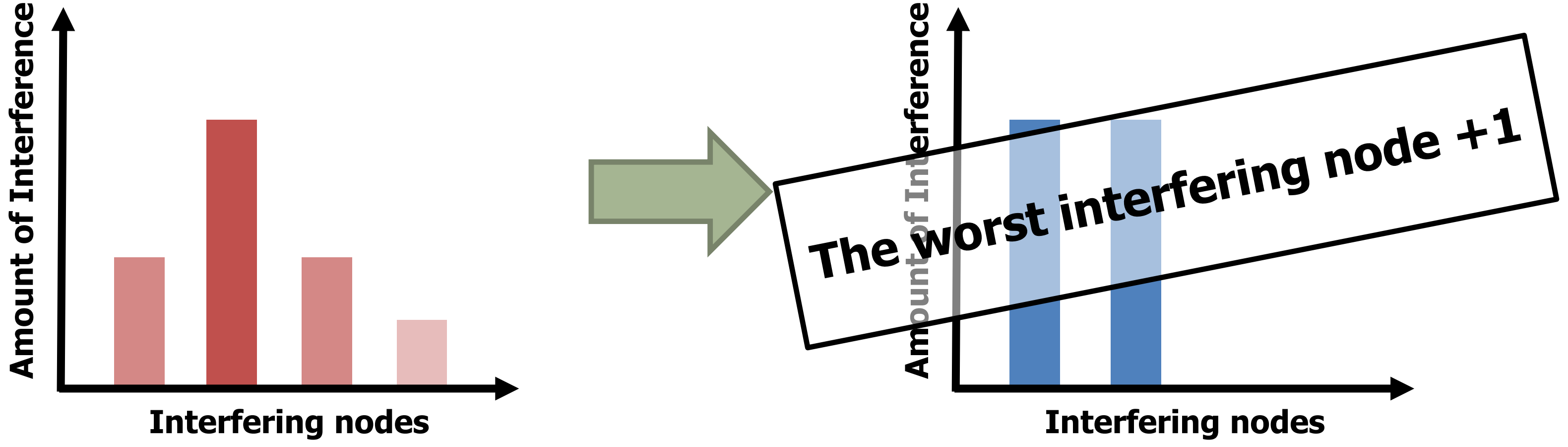
Conversion Policies

- 4 available conversion policies
  - **N max**
    - Considers only the worst interfering nodes
  - N+1 max
  - All max
  - Interpolate



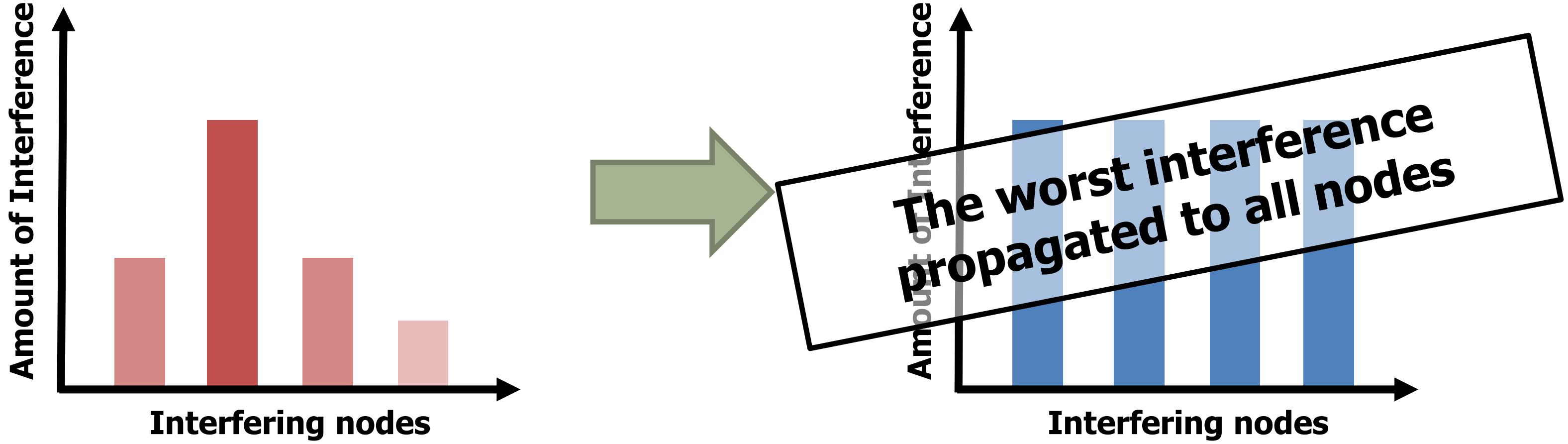
Convert Policies

- 4 available conversion policy
  - N max
  - **N+1 max**
    - Augments N max policy
    - The rest of interfering nodes are merged to the same worst pressure
  - All max
  - Interpolate



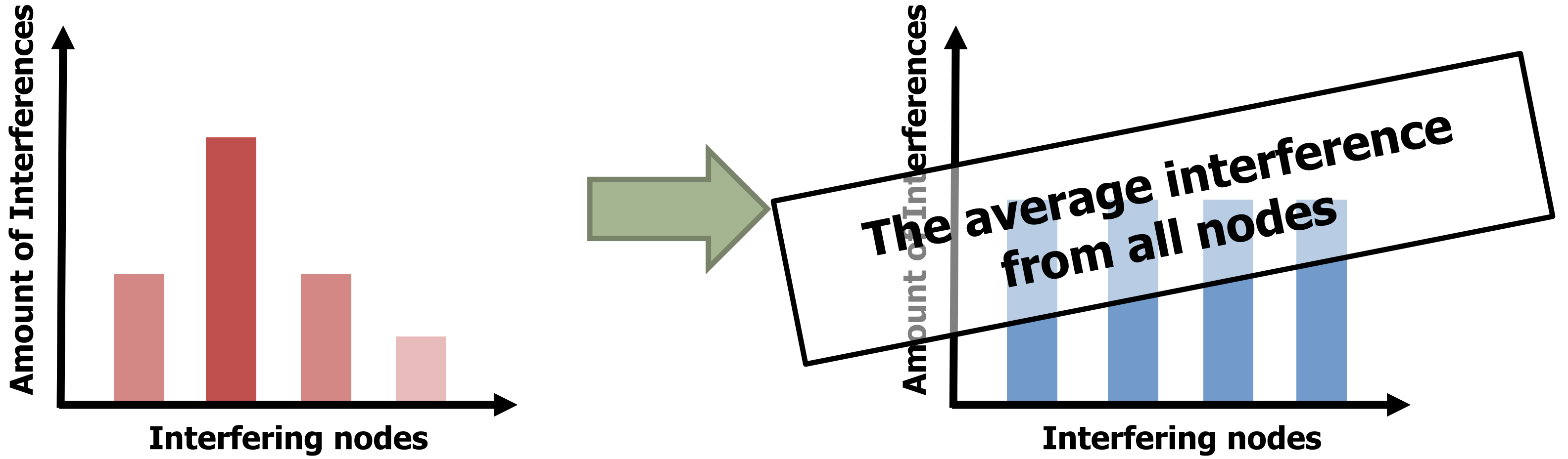
Convert Policies

- 4 available convert policies
  - N max
  - N+1 max
  - **All max**
    - The worst pressure propagates directly to all nodes
  - Interpolate



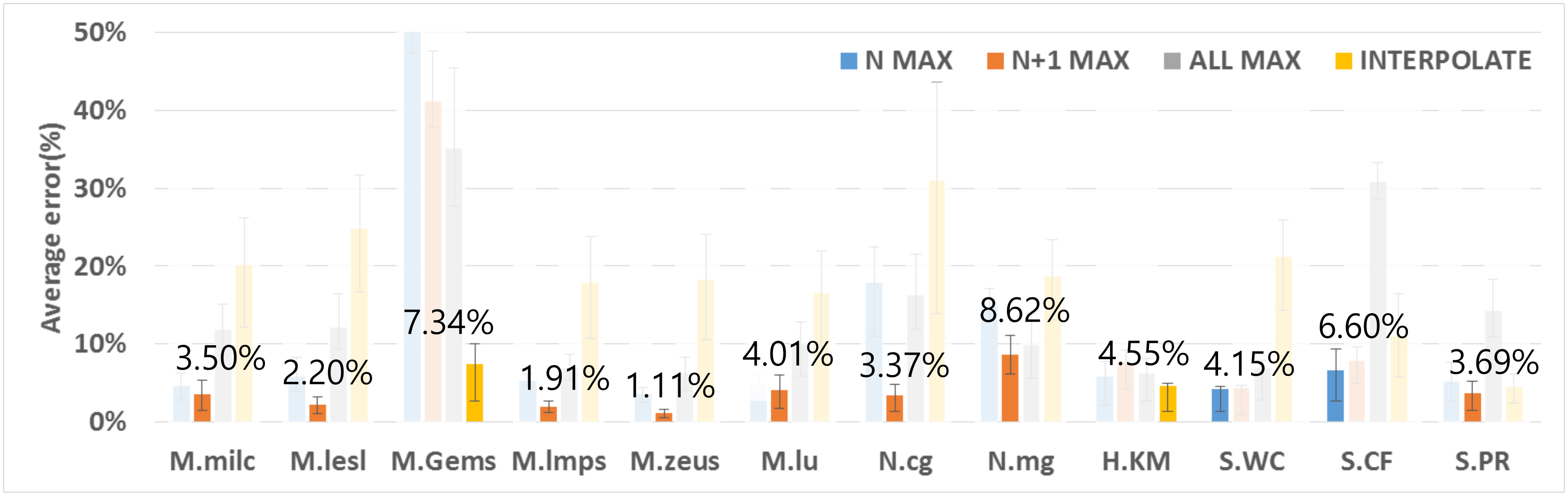
Convert Policies

- 4 available convert policies
  - N max
  - N+1 max
  - All max
  - **Interpolate**
    - Average interference from all nodes



Selecting Optimal Conversion Policy

- Evaluate 4 policies for each application



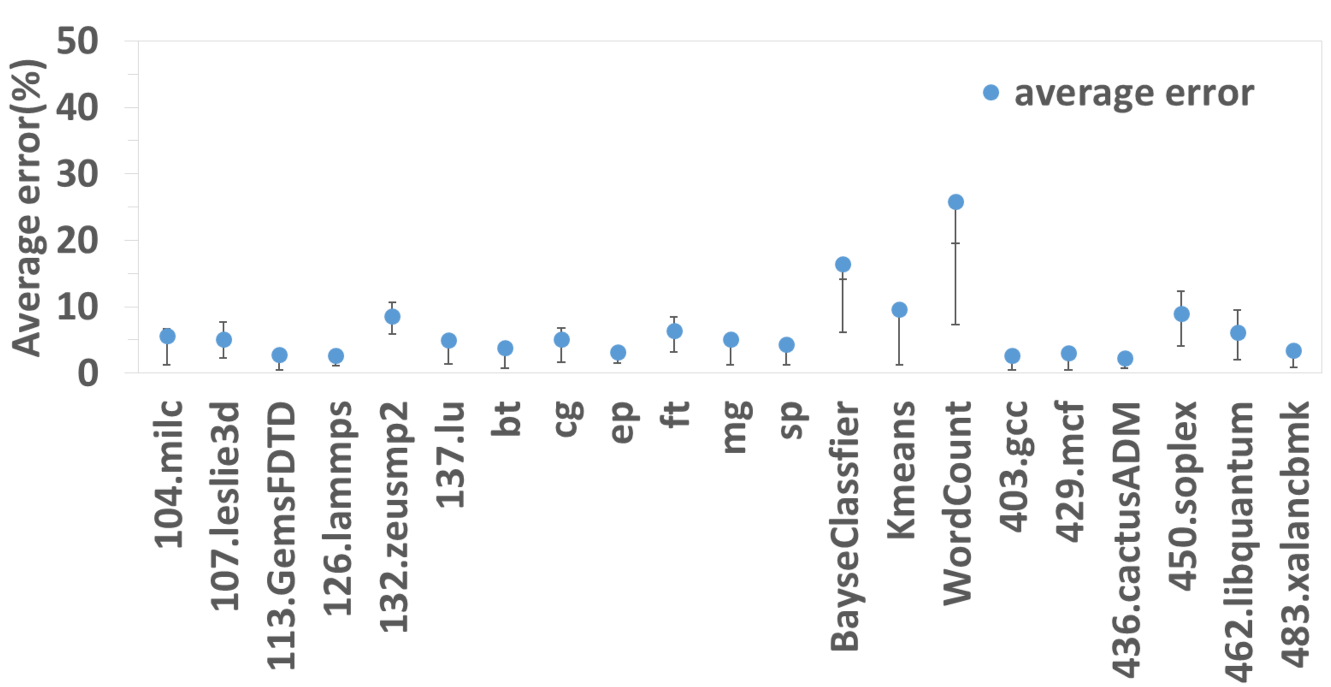
- Select the best policy for each application
- Achieve less than 9% average error

Performance Estimation Steps

- Building interference profile for each application
  1. Build interference propagation profile (binary-optimized profiling)
  2. Measure interference intensity generated from the application (pressure score)
  3. Find the best heterogeneity conversion policy (random sampling)
- Estimating application execution time in a consolidated cluster
  1. For each node, find the interference intensity from the co-runner
  2. Apply the heterogeneity conversion policy, and find a hypothetical run with homogeneous interference
  3. Use the propagation profile to estimate the final execution time

Validation Results

- All possible pairwise combinations of workloads in consolidated runs
- The average error for each application against all the other applications as the co-runner
- Most of the workloads have less than 10% errors



Two Case Studies

- Placement for performance
  - Maximize the overall cluster throughput
  - Selected 10 workload combinations
  - Use simulated annealing(*SA*) as placement algorithm

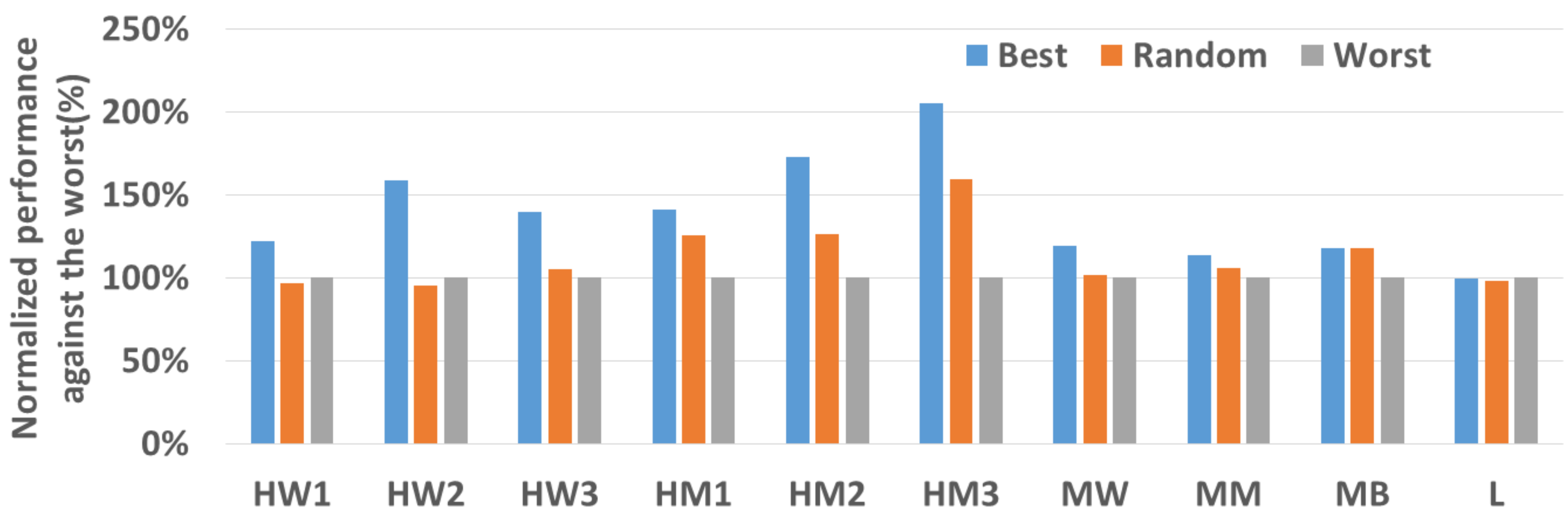
- QoS-Aware placement
  - 1 target workload + 3 different co-runners
  - Provide QoS guarantee for the target workload
    - Compare to zero interference run
  - Use *SA* under the QoS Constraints as placement algorithm

Two Case Studies

- **Placement for performance**
  - **Maximize the overall cluster throughput**
  - **Selected 10 workload combinations**
  - **Use simulated annealing(*SA*) as placement algorithm**

- QoS-Aware placement
  - 1 target workload + 3 different co-runner
  - Provide QoS guarantee for the target workload
    - Compare to zero interference run
  - Use *SA* under the QoS Constraints as placement algorithm

Placement Results

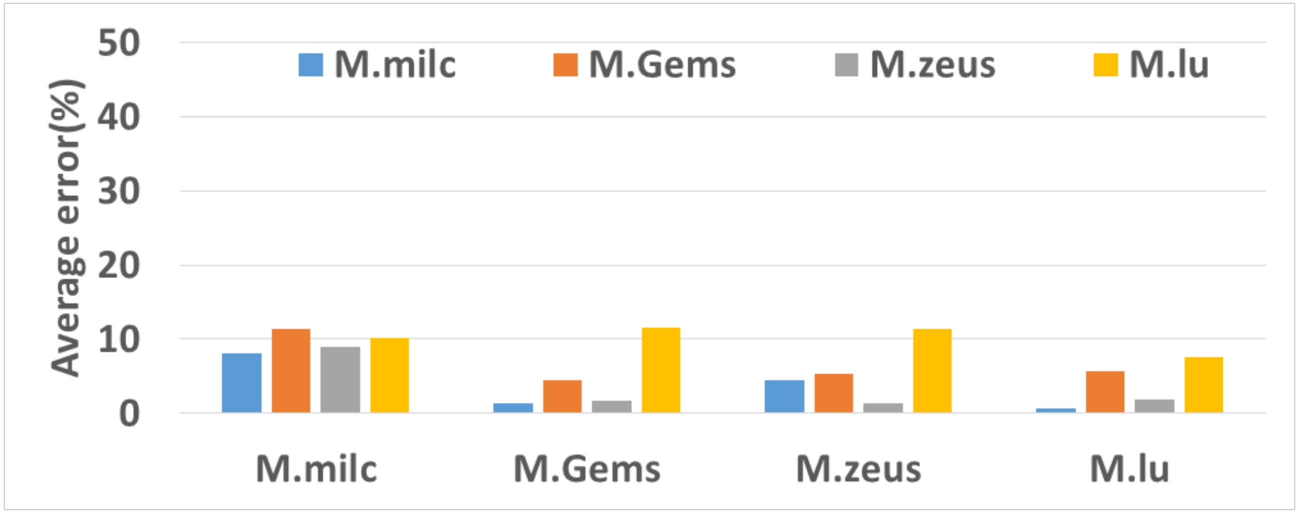


- Best : the best placement based on performance estimation
- Random : Average result of 5 random placements
- Worst : the worst placement based on performance estimation

Results from Amazon EC2

- Validation for larger systems

Workload	Best Policy	Avg. error(%)	Std. dev.
M.milc	N+1 max	12.01	7.27
M.Gems	N+1 max	11.49	6.28
M.zeus	ALL max	6.40	4.52
M.lu	N max	5.28	4.36



Conclusion

- Proposed a profiling-based interference estimation for distributed applications
  - Extended the *Bubble-Up* technique
- Per-application interference profile
  - Pressure score + propagation profile + heterogeneity conversion

- Limitation 1: Static profiling
  - Assume a priori knowledge of each application
  - Cannot reflect dynamic changes
- Limitation 2: Pairwise interaction
  - Up-to two applications can be co-located on each node