Programming for Xeon Phi using Cache Line awareness

Sabela Ramos
University of A Coruña, Spain
Torsten Hoefler
SPCL- ETH Zürich, Switzerland

What's unique about my tuning work

- **Shared memory apps in Xeon Phi are based on cache line transfers**
  - Cache coherency is one of the challenges of scalability in many-cores
- **GOALS:**
  - Turn the characterization of these transfers into a model.
  - Analyze codes in terms of cache line transfers.
  - Applicable to any Xeon Phi execution mode.
- **CHALLENGES:**
  - Characterize the cost.
    - DTDs made our life easier by providing very homogeneous latencies.
    - We developed a set of benchmarks to measure cache transfers.
  - Variability: line stealing.
    - Threads reading and writing to the same lines.
    - Min-Max models: estimate the cost of the best and the worst case.

Analyzing codes in terms of CL transfers

```
void Sync (int root, int my_id, int value, int *flag){
  if(my_id == root){
    *flag = value;
  }
  else{
    while (*flag!=value);
  }
}
```

Write flag

Read flag

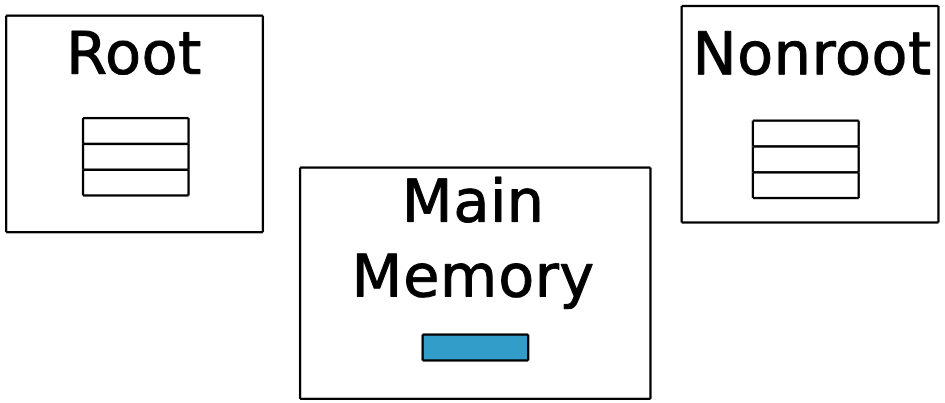
- ***In this example:***
  - ***we consider that variables like*** `my_id, value` ***and*** `root` ***are in registers.***
  - ***we assume that flag is in root's cache at the beginning.***
  - ***there is only two threads: root and nonroot.***

Analyzing codes in terms of CL transfers

```
void Sync (int root, int my_id, int value, int *flag){
  if(my_id == root){
    *flag = value;
  }
  else{
    while (*flag!=value);
  }
}
```

Write flag

Read flag

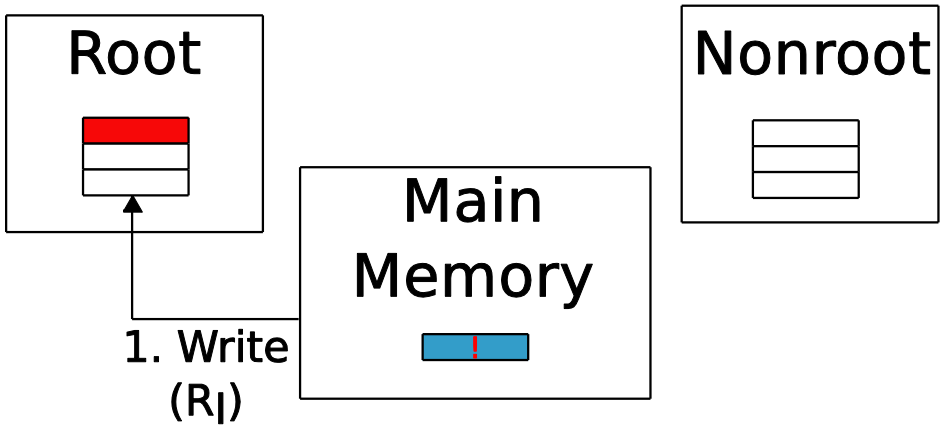


Analyzing codes in terms of CL transfers

```
void Sync (int root, int my_id, int value, int *flag){
  if(my_id == root){
    *flag = value;
  }
  else{
    while (*flag!=value);
  }
}
```

Write flag

Read flag

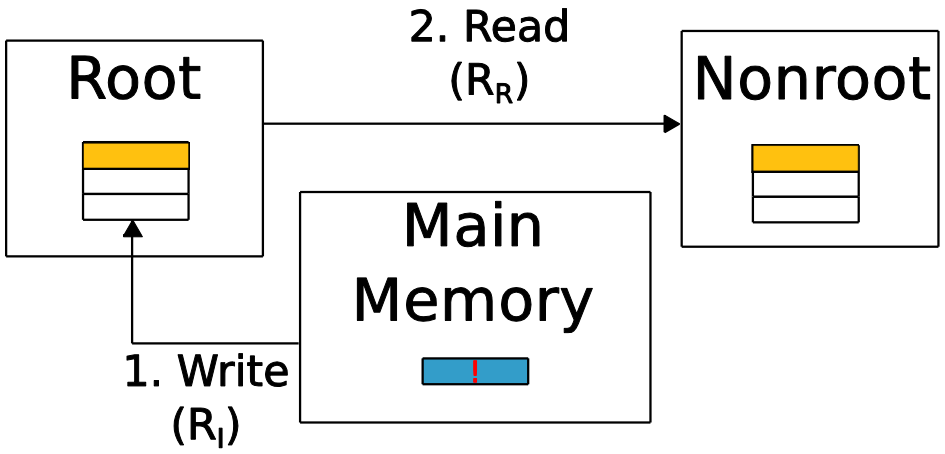


Analyzing codes in terms of CL transfers

```
void Sync (int root, int my_id, int value, int *flag){
  if(my_id == root){
    *flag = value;
  }
  else{
    while (*flag!=value);
  }
}
```

Write flag

Read flag

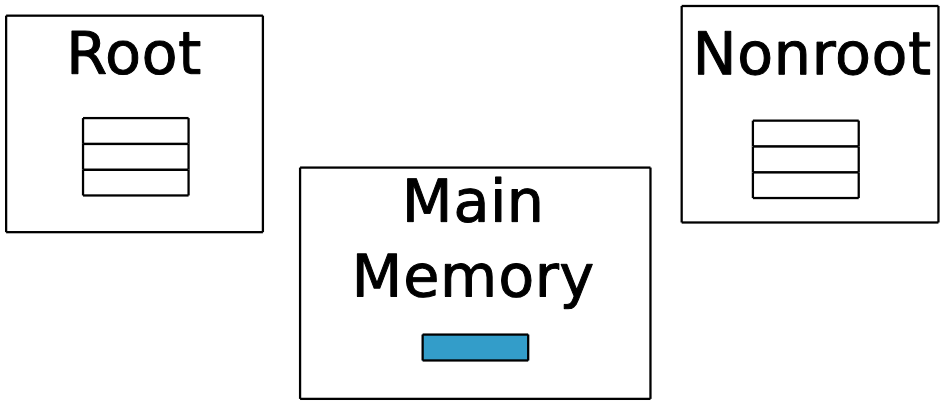


Analyzing codes in terms of CL transfers

```
void Sync (int root, int my_id, int value, int *flag){
  if(my_id == root){
    *flag = value;
  }
  else{
    while (*flag!=value);
  }
}
```

Write flag

Read flag

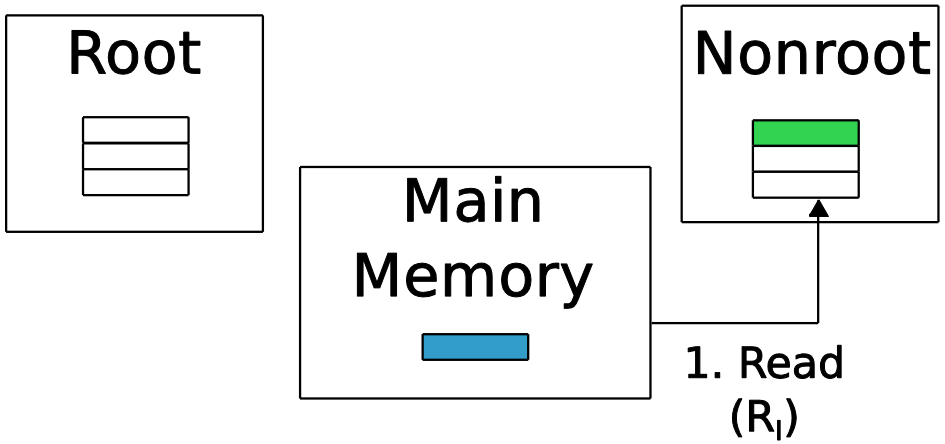


Analyzing codes in terms of CL transfers

```
void Sync (int root, int my_id, int value, int *flag){
  if(my_id == root){
    *flag = value;
  }
  else{
    while (*flag!=value);
  }
}
```

Write flag

Read flag

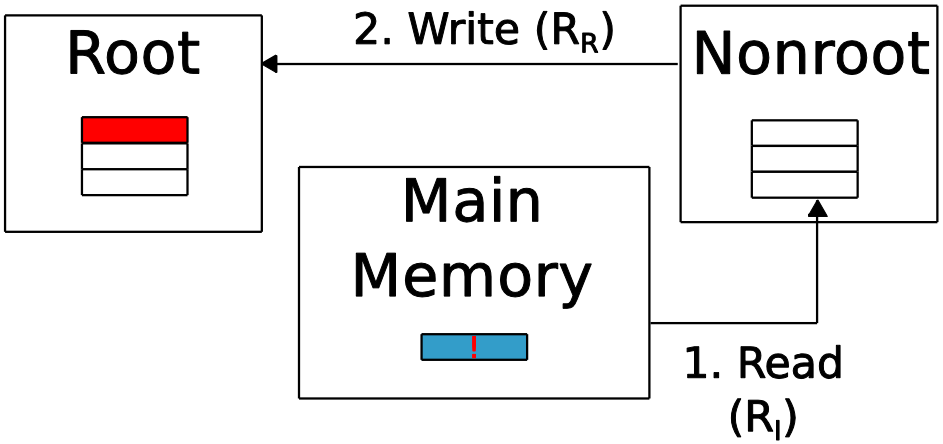


Analyzing codes in terms of CL transfers

```
void Sync (int root, int my_id, int value, int *flag){
  if(my_id == root){
    *flag = value;
  }
  else{
    while (*flag!=value);
  }
}
```

Write flag

Read flag

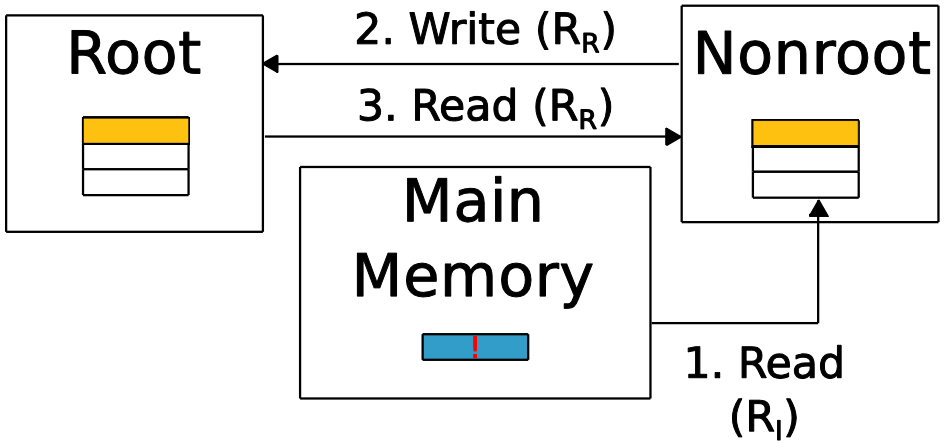


Analyzing codes in terms of CL transfers

```
void Sync (int root, int my_id, int value, int *flag){
  if(my_id == root){
    *flag = value;
  }
  else{
    while (*flag!=value);
  }
}
```

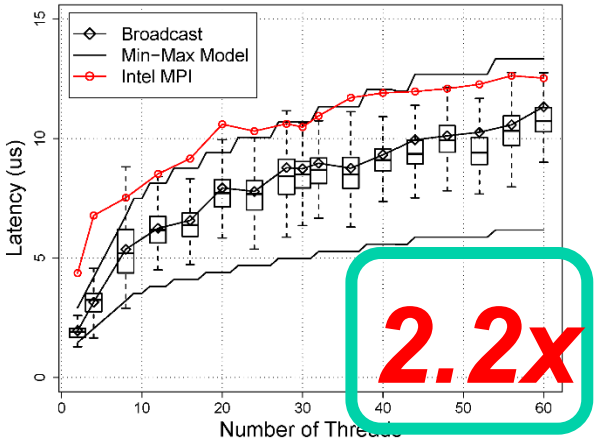
Write flag

Read flag



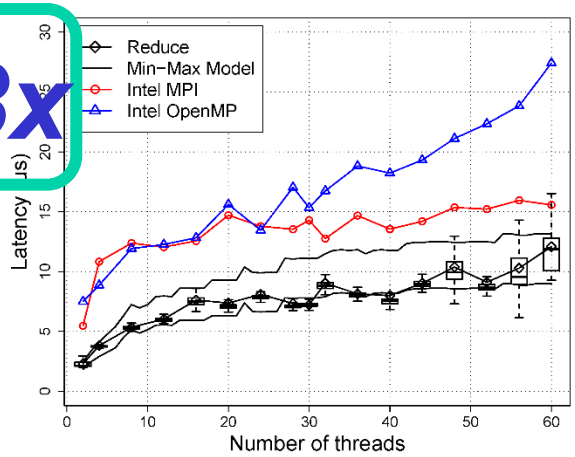
Performance

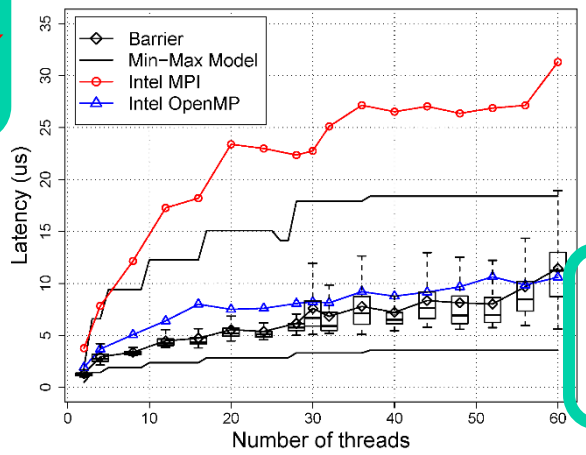
- **Methodology for designing and optimizing algorithms:**
  1. Express the algorithm in terms of cache transfers.
  2. Analyze the cost of each transfer and possible sources of variability.
  3. If there is any parameter, find the values that minimize the cost of the algorithm



2.2x

2.8x **3.3x**





4.3x **1.7x**

Speedup regarding
Intel MPI
Intel OpenMP

Insights

- **What we learned**
  - Cache line awareness enables deeper reasoning about performance and thread interaction.
  - The DTDs make distance between cores nearly irrelevant,
  - But cost of accessing other L2 slices is high:
    - Placement optimizations must focus on which threads share a core and on data locality, rather than on using specific cores.
  - The ring shows contention but almost no congestion:
    - Avoid a large number of threads reading the same data.
- **What we need to improve**
  - Automate the model derivation
  - Include probabilistic terms in the min-max models
- **Further readings:**
  - *Sabela Ramos, Torsten Hoefler*, "Modeling Communication in Cache-Coherent SMP Systems - A Case-Study with Xeon Phi", **HPDC'13, pp. 97-108**
  - Check our benchmarks in http://gac.des.udc.es/~sramos/xeon_phi_bench/xeon_phi_bench.html