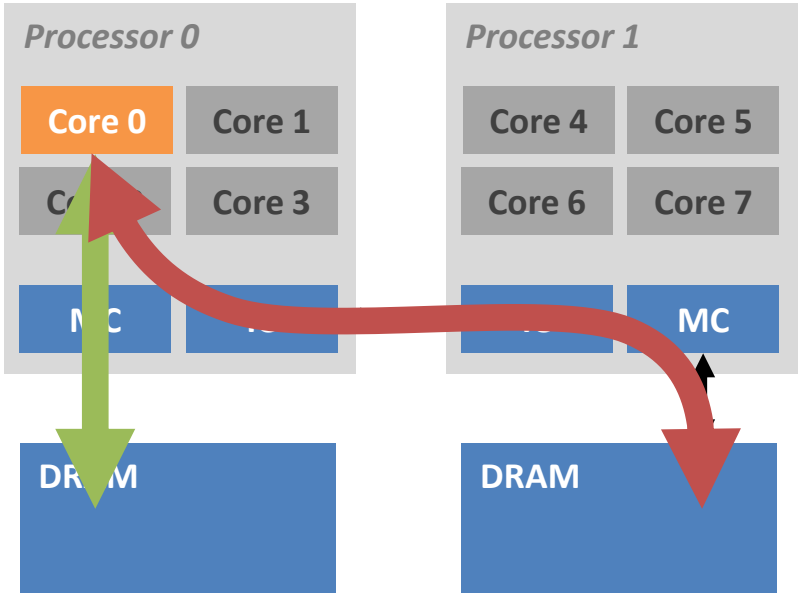
A Template Library to Integrate Thread Scheduling and Locality Management for NUMA Multiprocessors

Zoltán Majó
Thomas R. Gross

Computer Science Department
ETH Zurich, Switzerland

Non-uniform memory architecture



Local memory accesses

bandwidth: 10.1 GB/s

latency: 190 cycles

Remote memory accesses

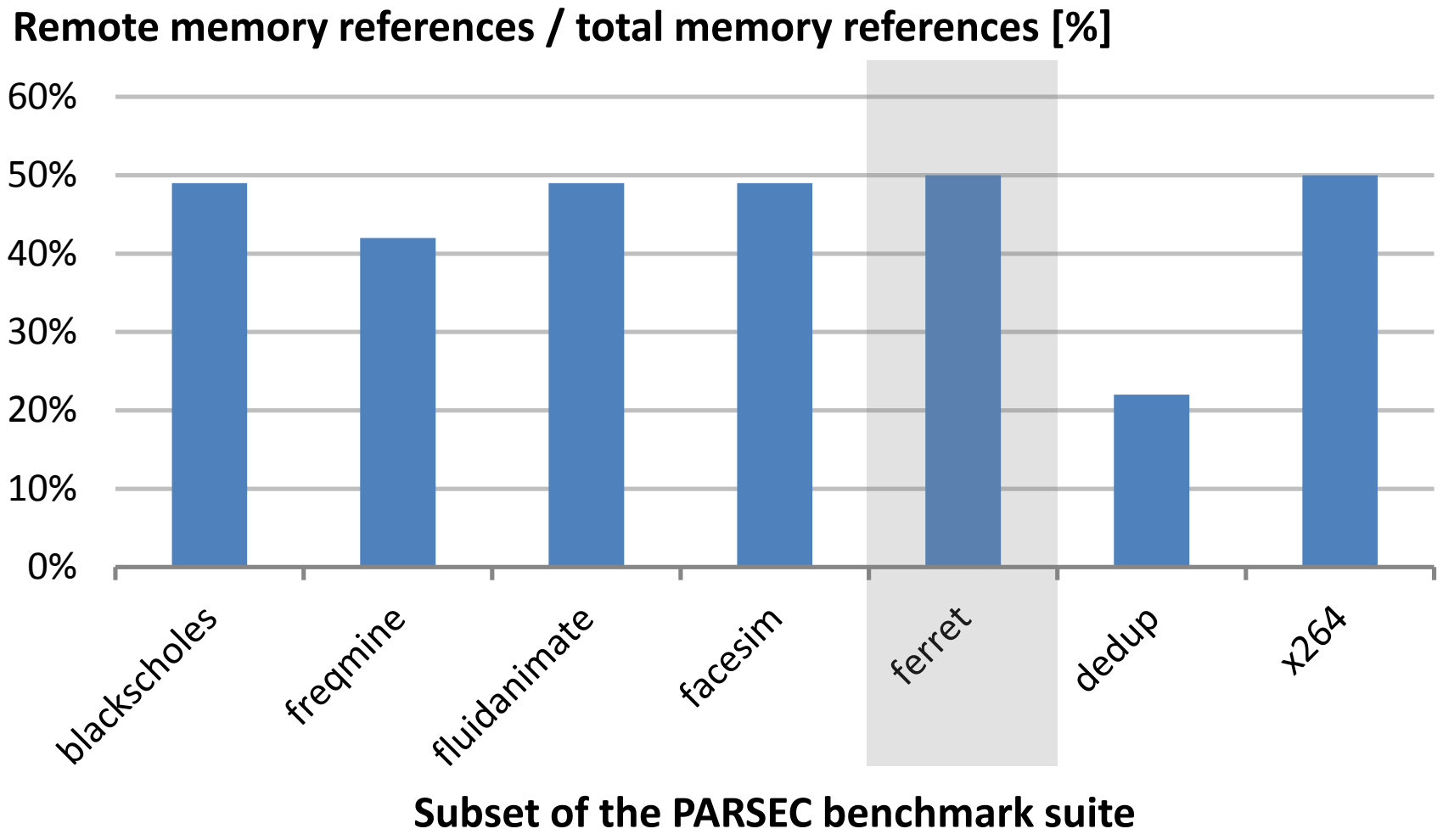
bandwidth: 6.3 GB/s

latency: 310 cycles

Key to good performance: ***data locality***

All data based on experimental evaluation of Intel Xeon 5500 (Hackenberg [MICRO '09], Molka [PACT '09])

Remote memory references (2 processors)

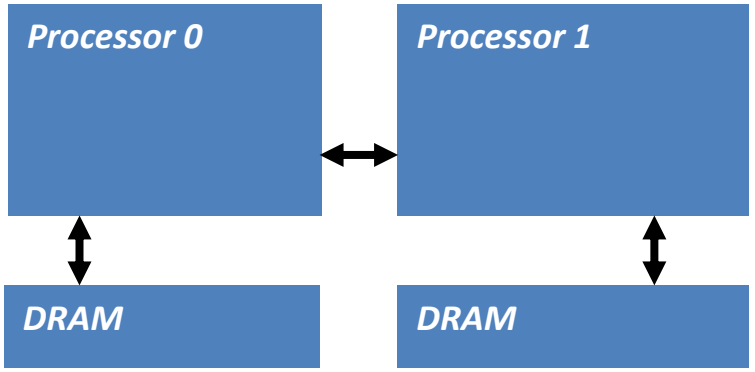


Outline

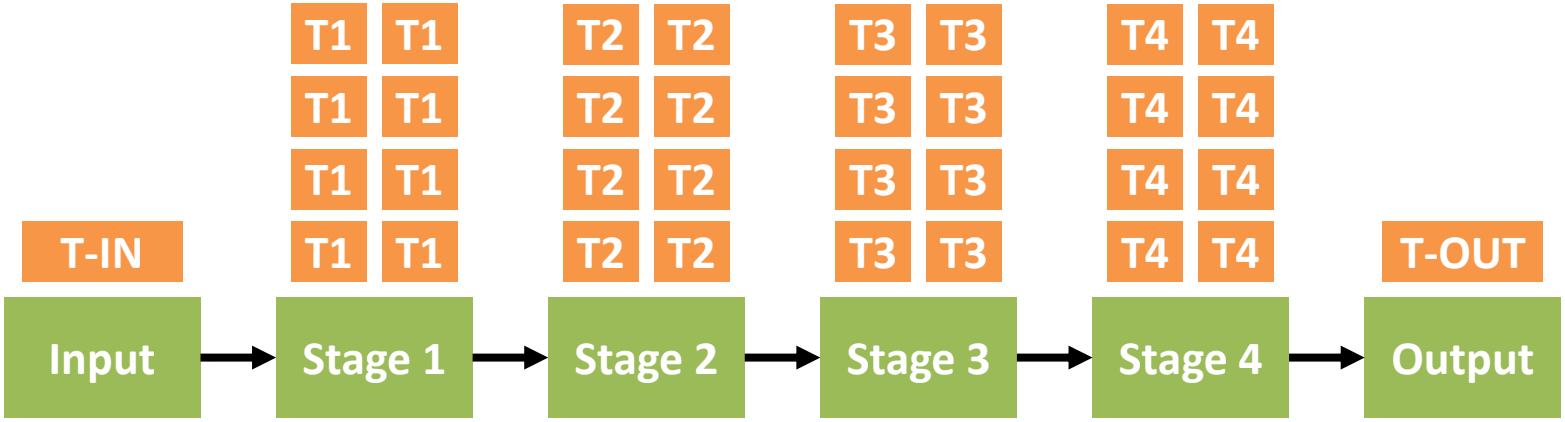
- **Introduction**
- **Ferret**
  - Memory behavior
  - Optimizing for data locality
  - Evaluation
- **Template library**
- **Concluding remarks**

Experimental setup

- **2-processor 8-core Intel Xeon**
  - Nehalem microarchitecture
  - Linux + perfmon2
  - First-touch page placement



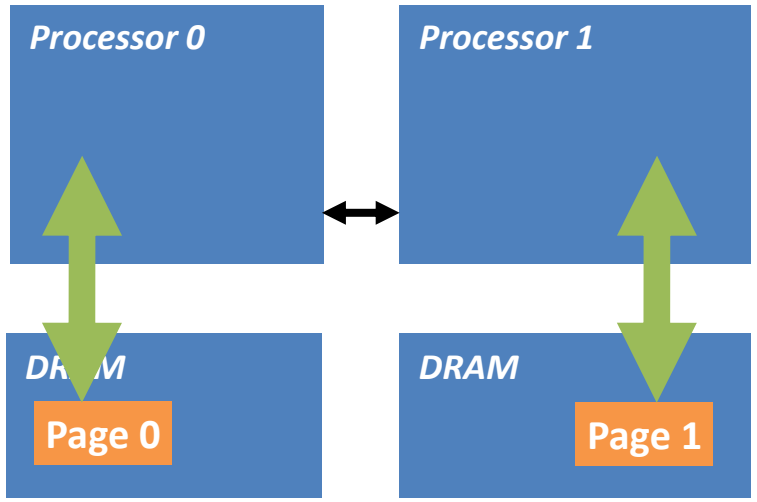
- **Ferret: pipeline parallelism**



Profiling memory accesses

- **Data address profiling**
  - Based on hardware-performance monitoring
  - Consider only heap accesses

Profile
Page 0 : 1000 accesses by Processor 0
Page 1 : 3000 accesses by Processor 1



Profiling memory accesses

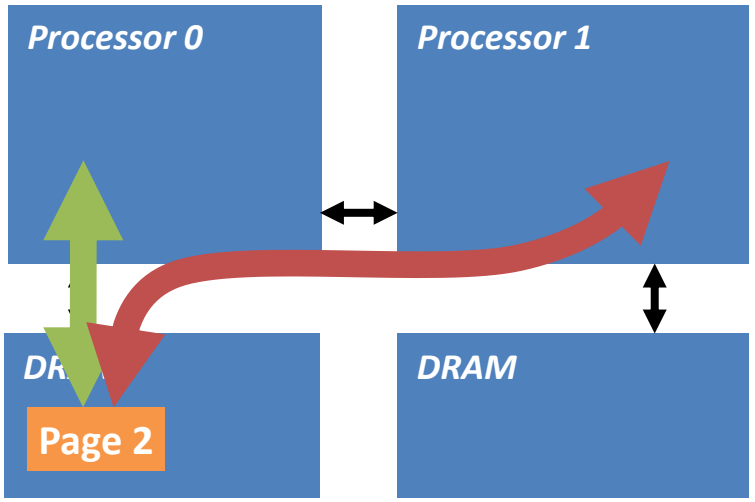
- **Data address profiling**
  - Based on hardware-performance monitoring
  - Consider only heap accesses

Profile

Page 0 : 1000 accesses by Processor 0

Page 1 : 3000 accesses by Processor 1

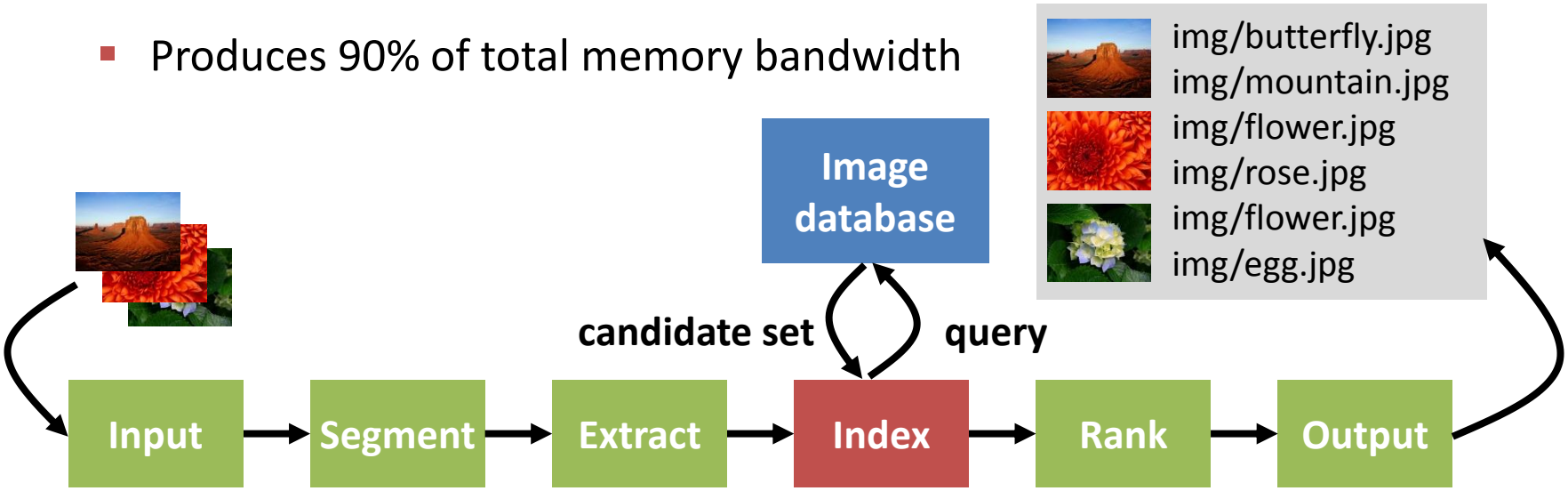
Page 2 : 4000 accesses by Processor 0
5000 accesses by Processor 1



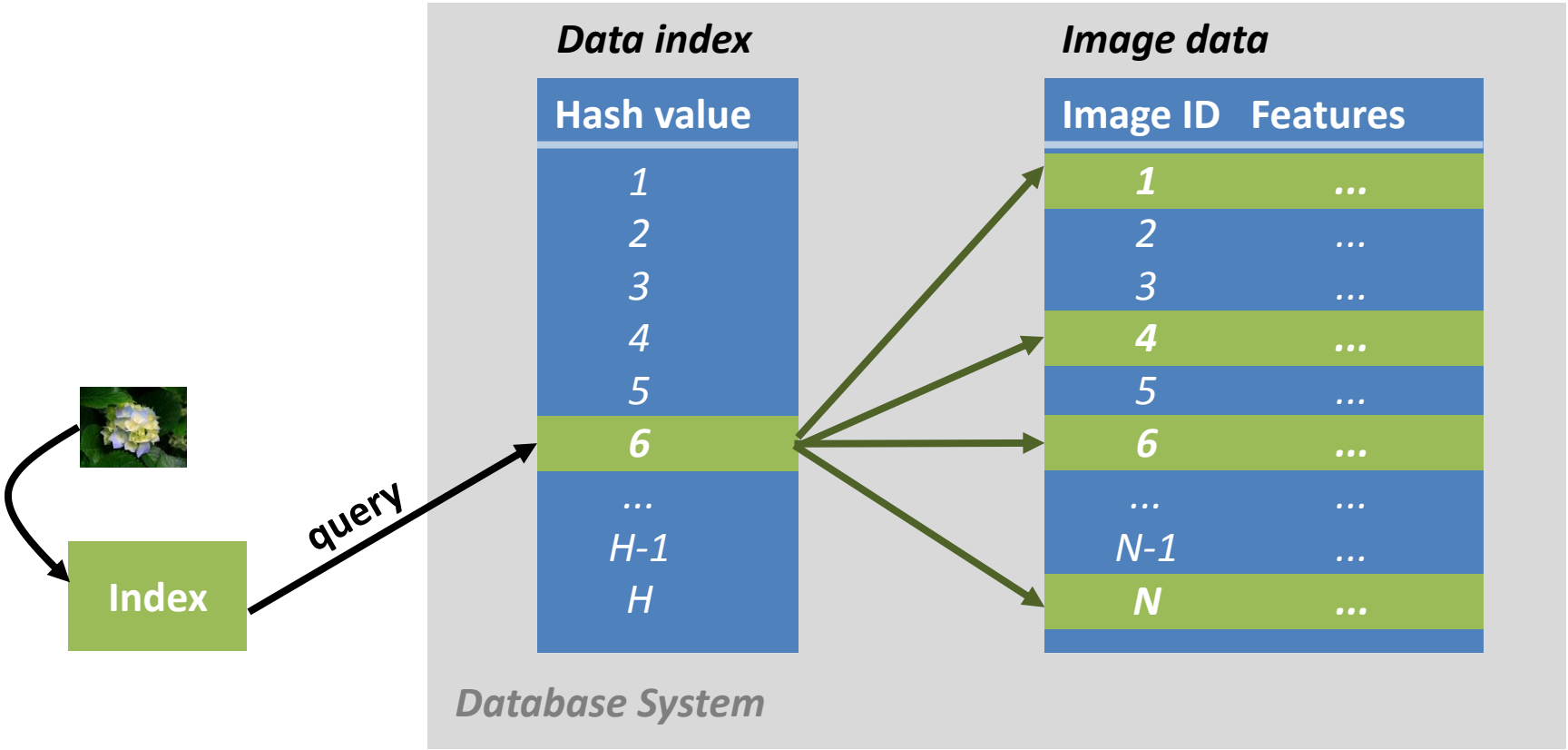
Ferret: ***41%*** of profiled pages inter-processor shared

Detailed look

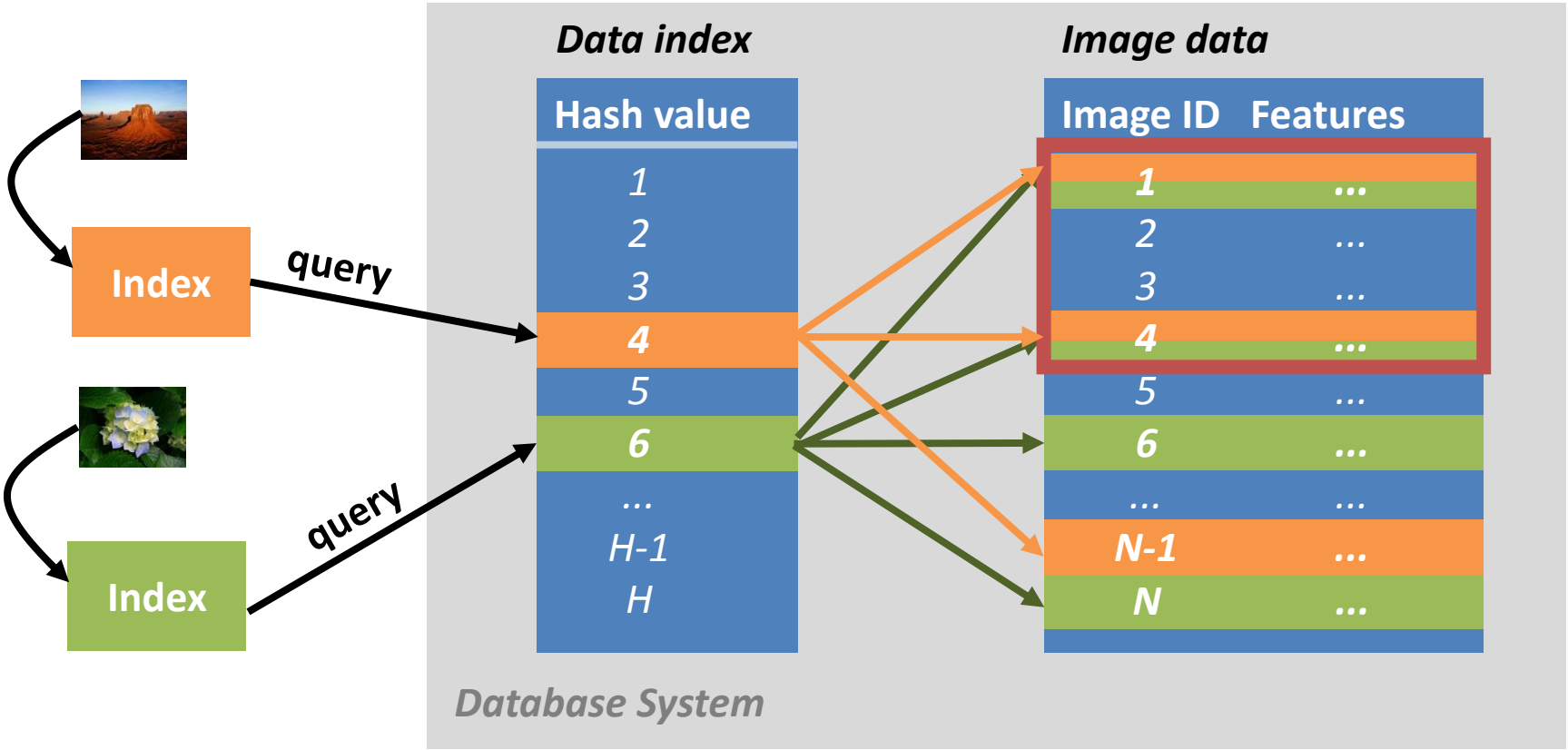
- **Ferret: similarity search of images**
  - Input: set of images
  - Output: list of images similar to input
- **Index stage:** ***most memory system intensive***
  - Produces 90% of total memory bandwidth



Locality-sensitive hashing



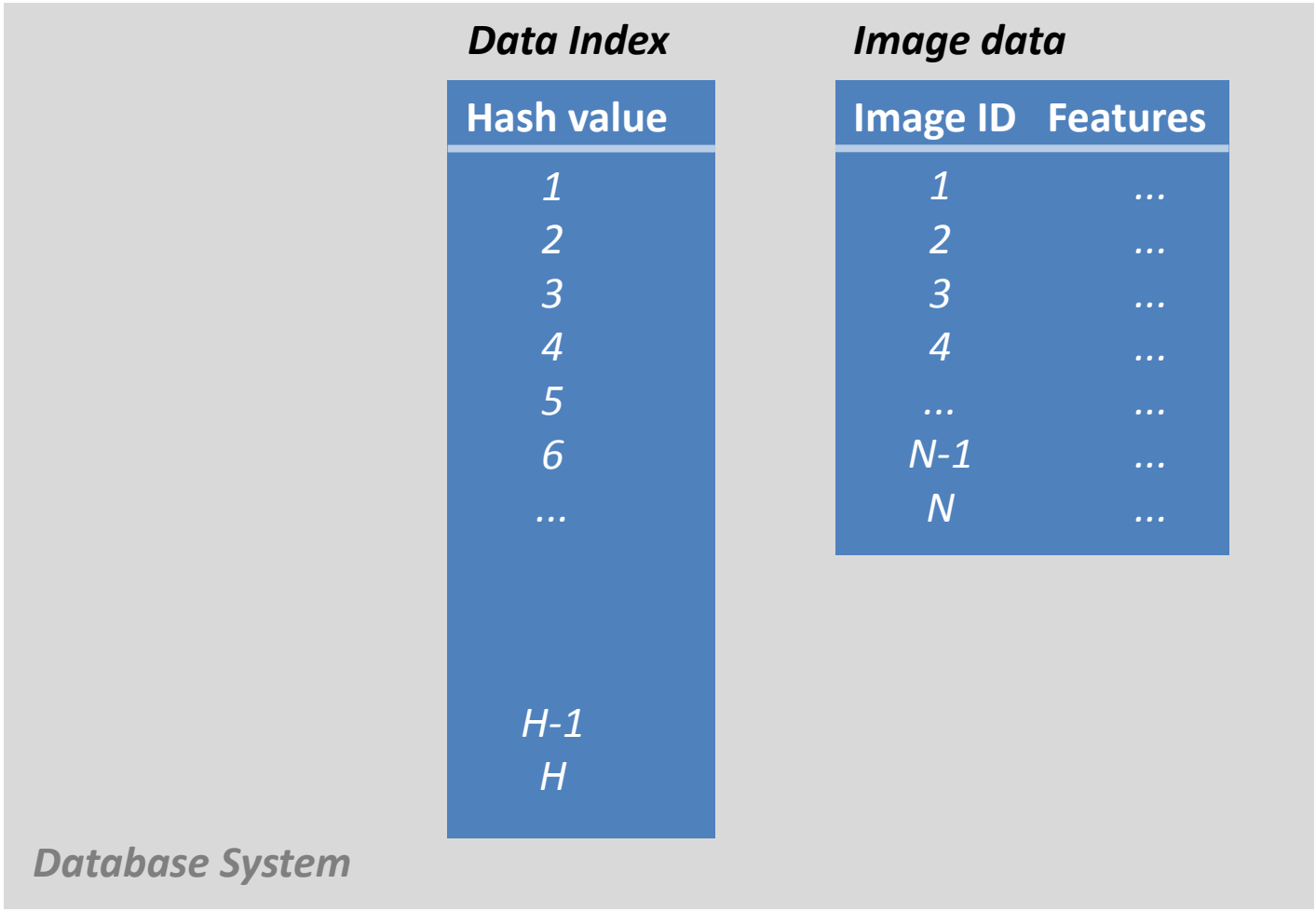
Locality-sensitive hashing



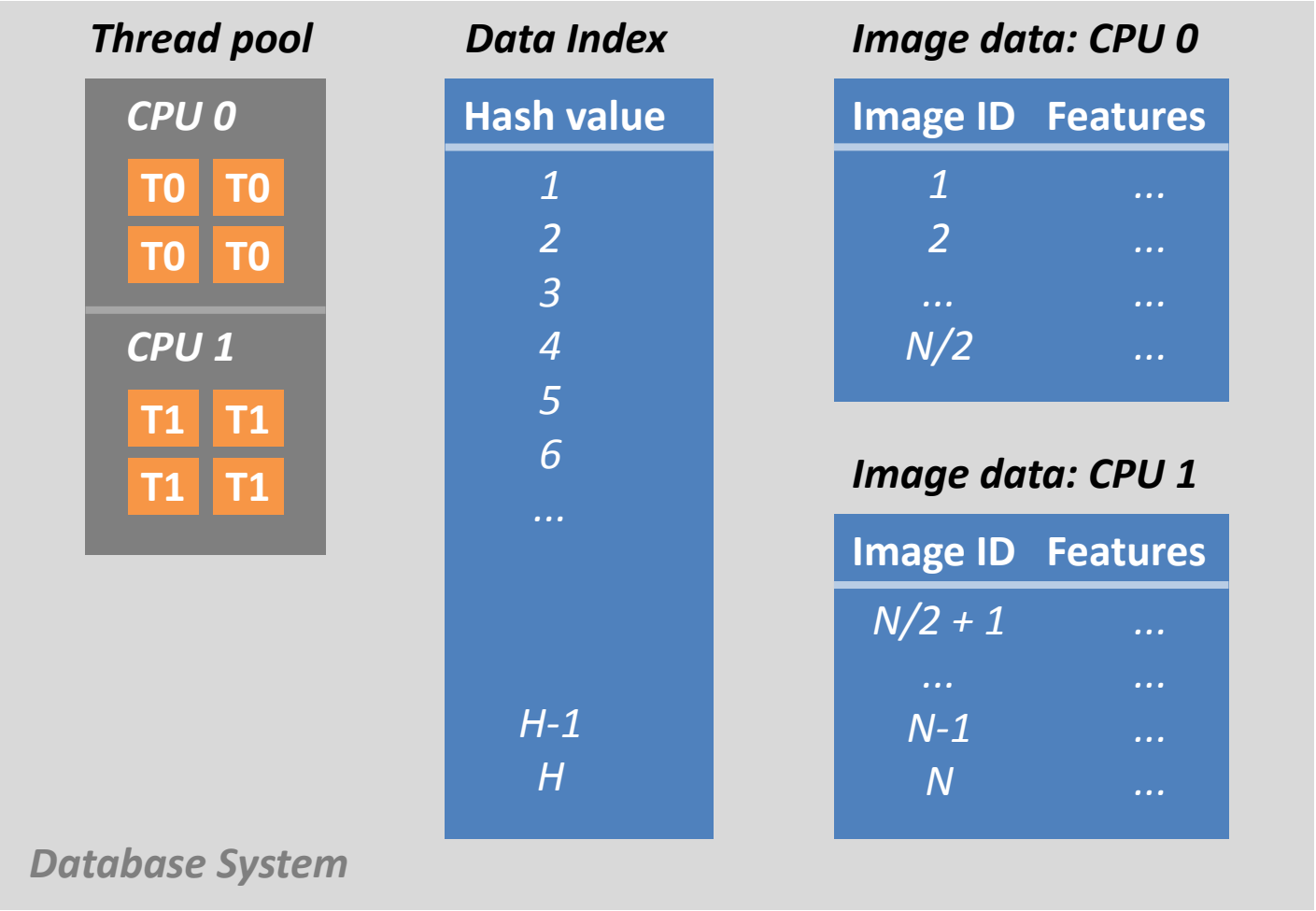
Bad data locality

- **Data accesses and parallelization *decoupled***
  - Database interface hides details about internal structure
  - All threads running the indexing phase access database
- **Optimizing for data locality**
  - Precise control of data and computation allocation ***within database***
  - Example: 2-processor system

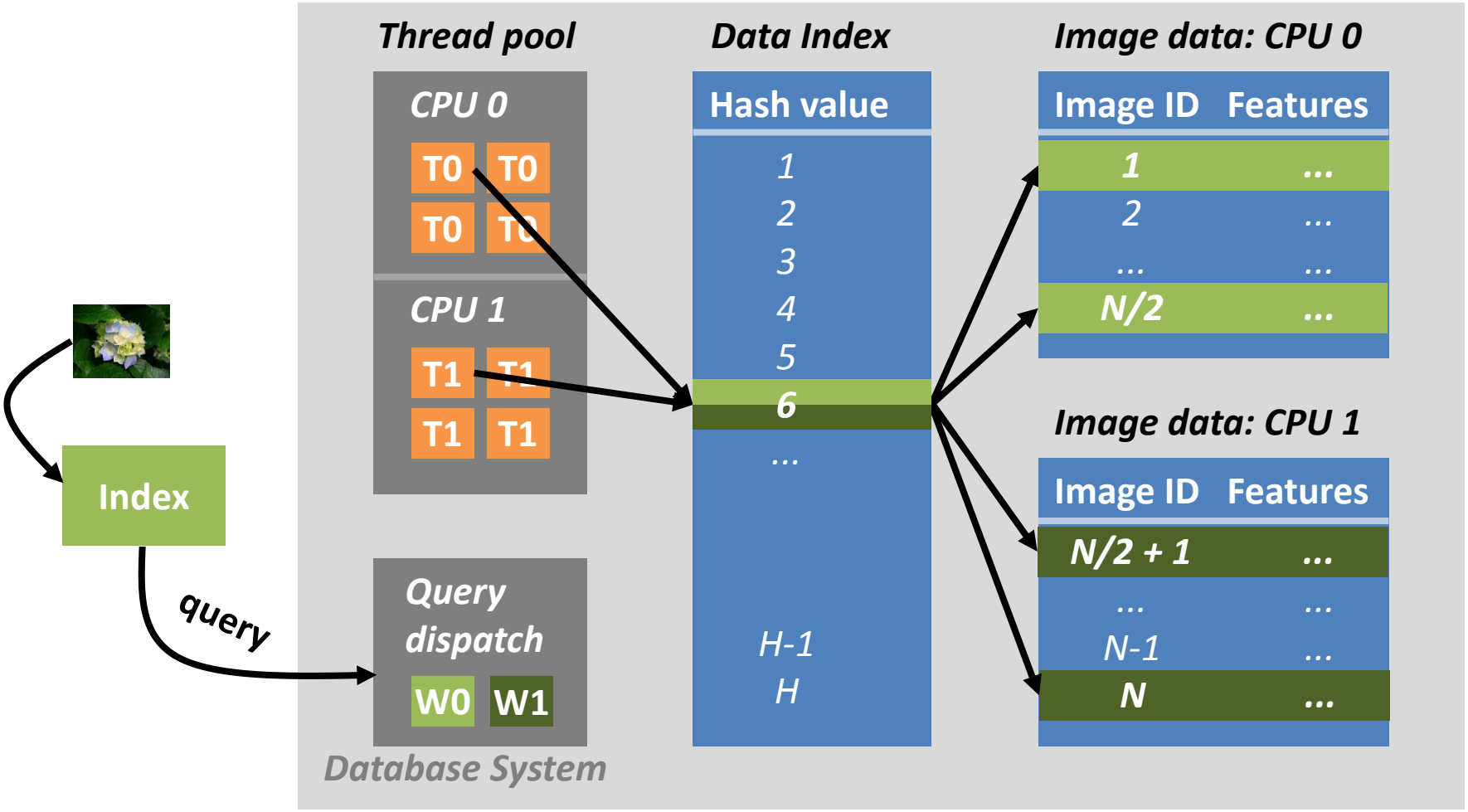
Optimizing for data locality



Optimizing for data locality



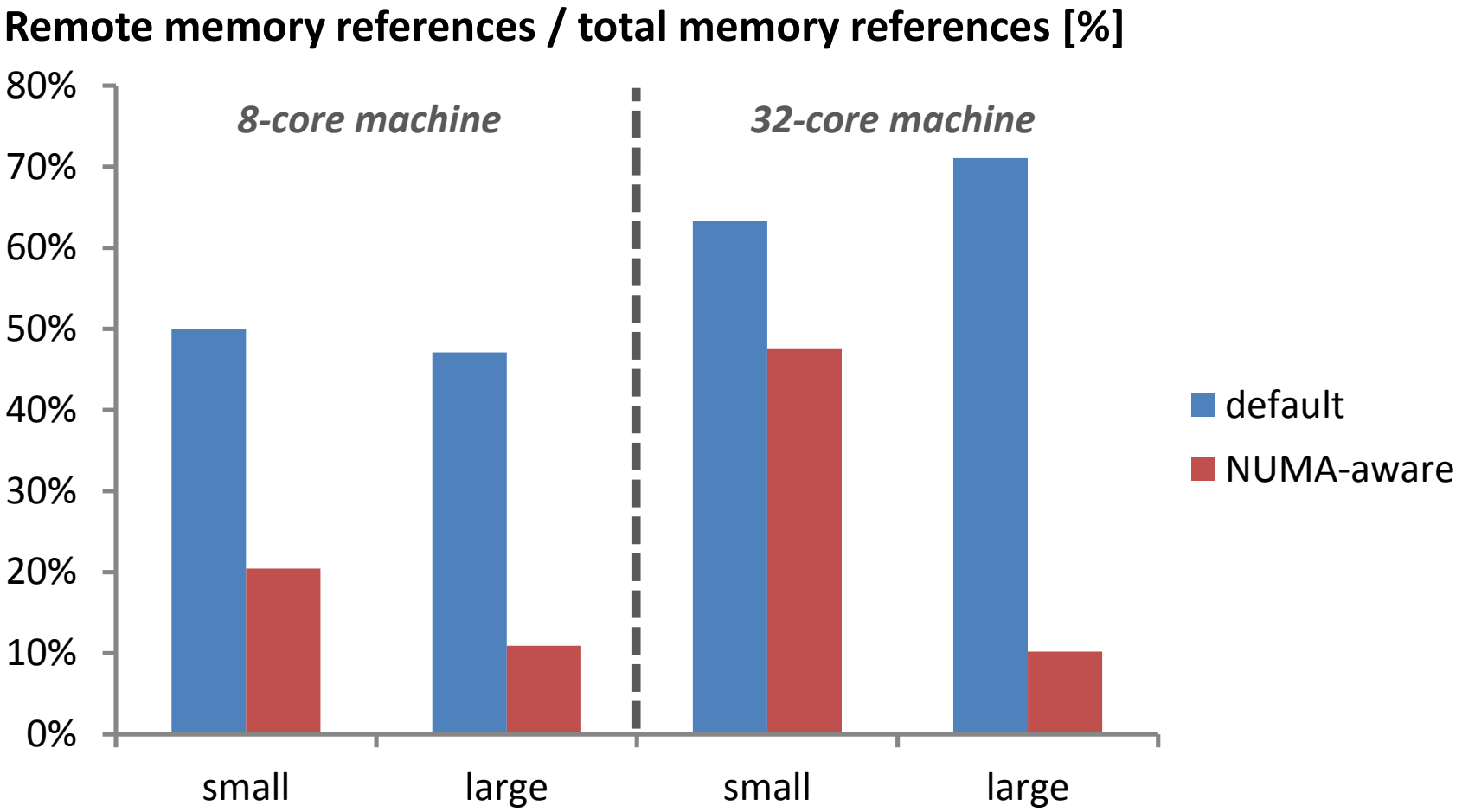
Optimizing for data locality



Evaluation

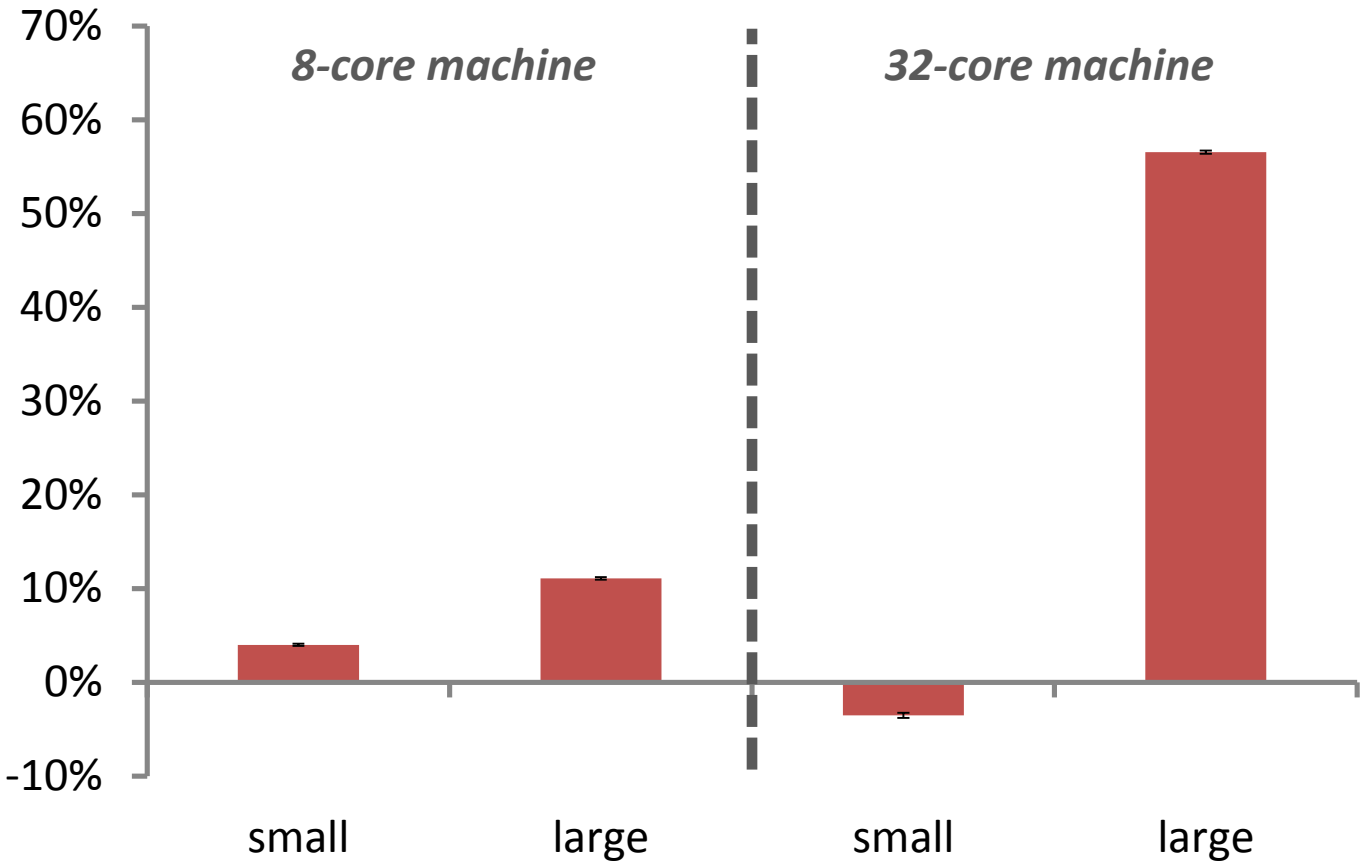
- **Two machines**
  - 2-processor **8-core** Intel Nehalem (Xeon E5520)
  - 4-processor **32-core** Intel Westmere (Xeon E7-4830)
- **Two image database sizes**
  - **small**: ~60 K images
  - **large**: ~744 K images
  - 3500 image queries in both cases
- **Compare two configurations**
  - **default**: first-touch page allocation, affinity scheduling
  - **NUMA-aware** memory allocation and computation scheduling

Data locality



Performance

Performance improvement of NUMA-aware over default [%]



Optimizing for data locality – summary

- **Optimizing data locality in ferret difficult**
  - System developer: internals of shared database must be understood
  - Database developer: system must be understood
- **Data structures often key to good NUMA performance**
- **Template library: basis for NUMA-aware data structures**

Template library

- **Base class**
  - Per-processor data allocation
  - Locality-aware task dispatch

```
SplittableData<Data,Result>
Data newAtProcessor(p:Processor)
Result dispatch(queryTask:Task)
```

Template library

```
abstract class SplittableData<Data,Result> {
     ThreadPool threadPool;
     Map<Processor,Data> map;

     Data newAtProcessor(Processor p) {
          // new Data instance at Processor p
          // record mapping Processor → Data
     }
     Result dispatch(Task queryTask) {
          List<Result> results;
          Data localData;
          for (Processor p : processors) {
               localData = map.get(p);
               threadPool.submit(queryTask, p, localData);
          }
          for (Processor p : processors)
               results.add(threadPool.get(p));
          return Result.merge(results);
     }
}
```

Concluding remarks

- **Some parallel programs NUMA-unfriendly *by construction***
- **Good performance: *programmer intervention required***
- **Template library to abstract low-level system details**

Thank you for your attention!