A Locality Model for the Real-Time Specification for Java

Abdul Haseeb Malik
Andy Wellings
Yang Chang

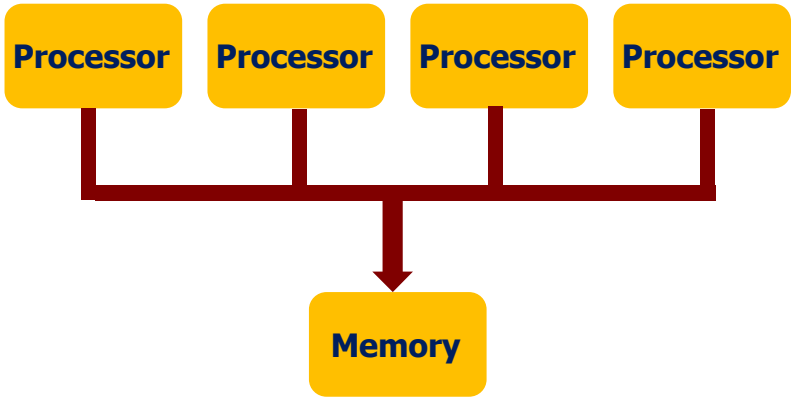
JTRES 2010
19-21 August,
Prague,
Czech Republic

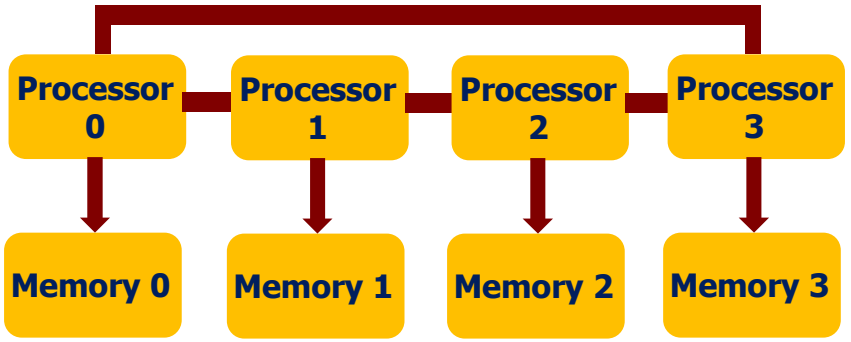
RTS York

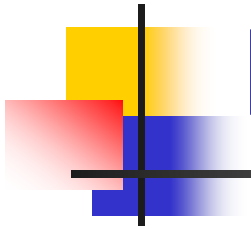
THE UNIVERSITY of York

Introduction

- Shift to multiprocessors
- UMA SMPs
  - Single address space
  - Cache coherence
  - Uniform memory access
- NUMA systems
  - Single address space
  - Global or partial cache coherence
  - Non-uniform memory access





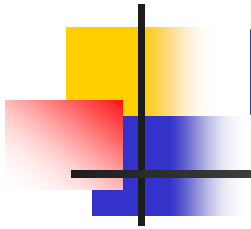


Problem

- Java applications experience unpredictable delays due to a large number of remote accesses.
- Remote accesses take considerably longer than local accesses.
- The application cannot differentiate between local and remote accesses because
  - NUMA architecture is hidden from the application.
  - Operating system manages allocation policies.
  - Application is unaware of these allocation policies.

Related Work

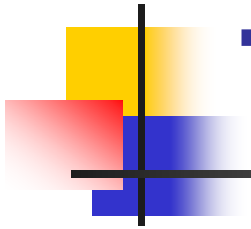
- High performance computing e.g. X10, Fortress, Chapel etc.
  - Representation of the architecture.
  - Grouping of tasks and objects.
  - Programmers can explicitly allocate objects on specific memory areas.



Existing Support

1. AffinitySet class
   - Threads can be allocated on specific processors.
2. Physical memory framework
   - Can be used to create physical memory areas on specific nodes.

- Both can be used to allocate threads and objects individually on desired nodes.

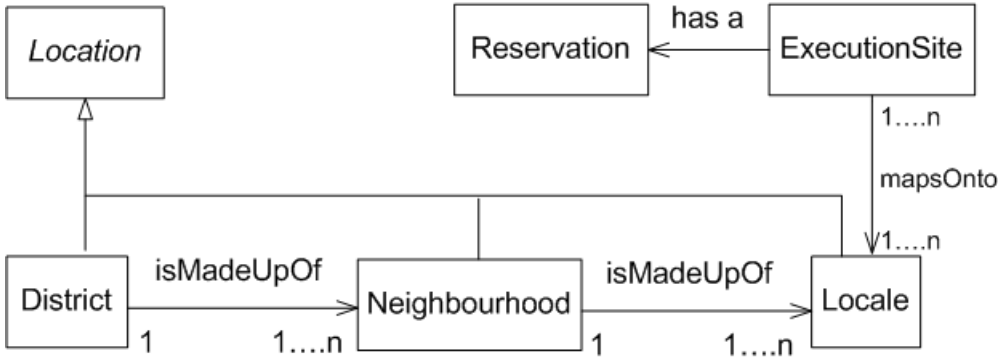


The Locality Model

- Introduces new abstractions which
  - Provide visibility into the system architecture.
  - Threads and objects grouped together.
  - Groups allocated
    - Statically by the programmer
    - Dynamically at runtime

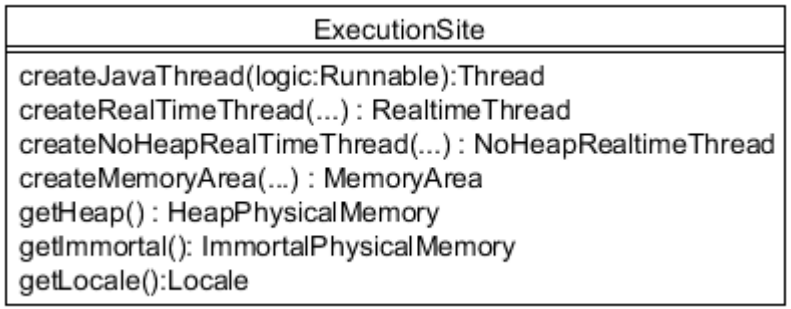
The Locality Model

- Locations: collection of processors, memory banks and devices.
  - Locale: logical representation of SMP.
  - Neighbourhood: logical representation of cc-NUMA
    - RTJVMs mapped on neighbourhoods.
  - District: logical representation of NUMA.
  - Execution site(ES): capable of executing an RTSJ program.

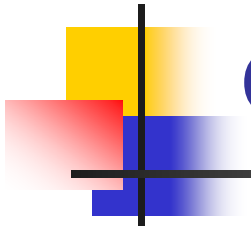


Execution Site

- Execution site created for more predictable behaviour.
- Has a heap, immortal memory and backing store for scoped memory areas.
- Factory methods to create threads.
- Factory methods to create scoped memory areas.



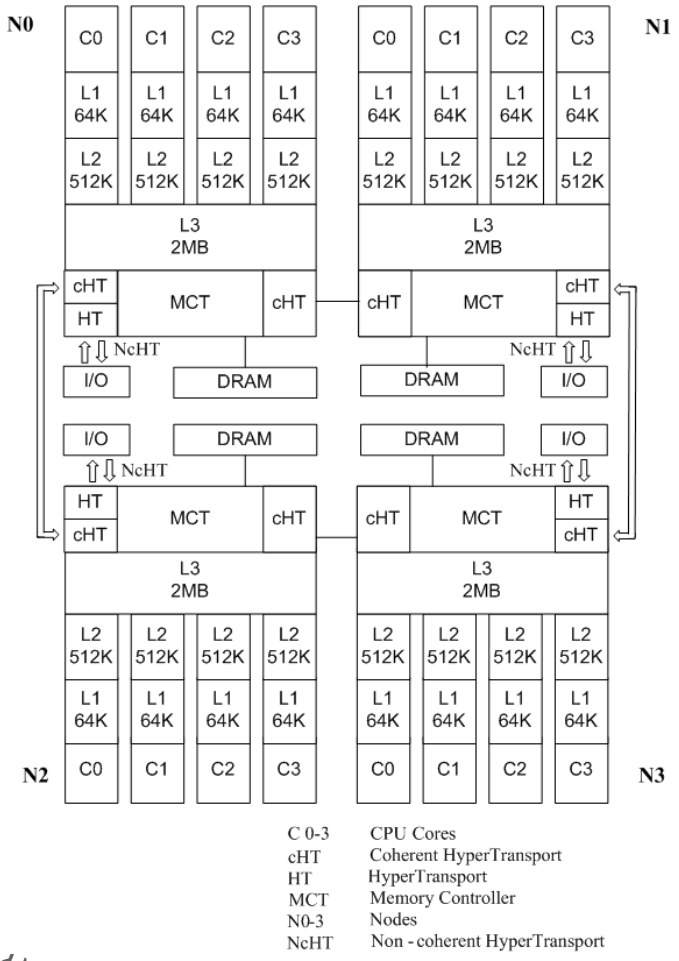
Creation and Mapping of Execution Sites



- Factory methods in the neighbourhood class
- Static mapping
  - forced by the input of the programmer.
  - application is not portable.
- Dynamic mapping
  - based on *reservations.*
  - resource requirements requested for each execution site in the form of *reservation parameters.*
  - mapped by the runtime based on the requirements of the execution site.

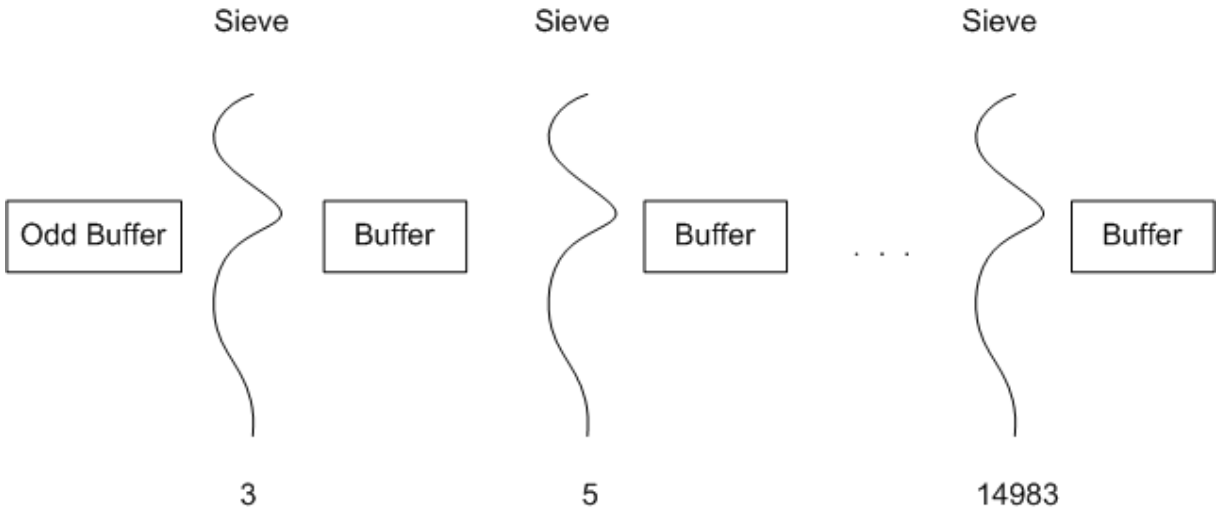
Prototype

- Prototype based on jRate over linux.
- Extensions made to the jRate runtime library.
  - memory areas in ES created using the NUMA API.
  - threads created inherit cpu affinity of the execution site.
- 16 processor cc-NUMA system based on AMD opteron.



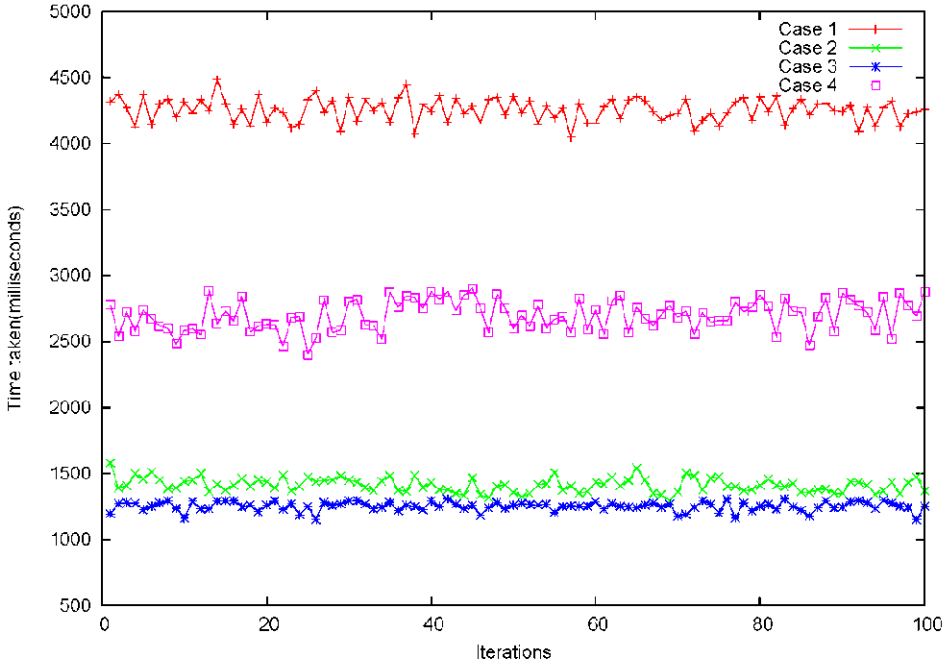
Experiment: Sieve of Eratosthenes

- Highly parallel algorithm
  - 1 thread for each prime number.
  - For all prime numbers< 15000 ~ 1754 threads.
  - Experiment will measure execution times for these 1754 threads using the locality model.



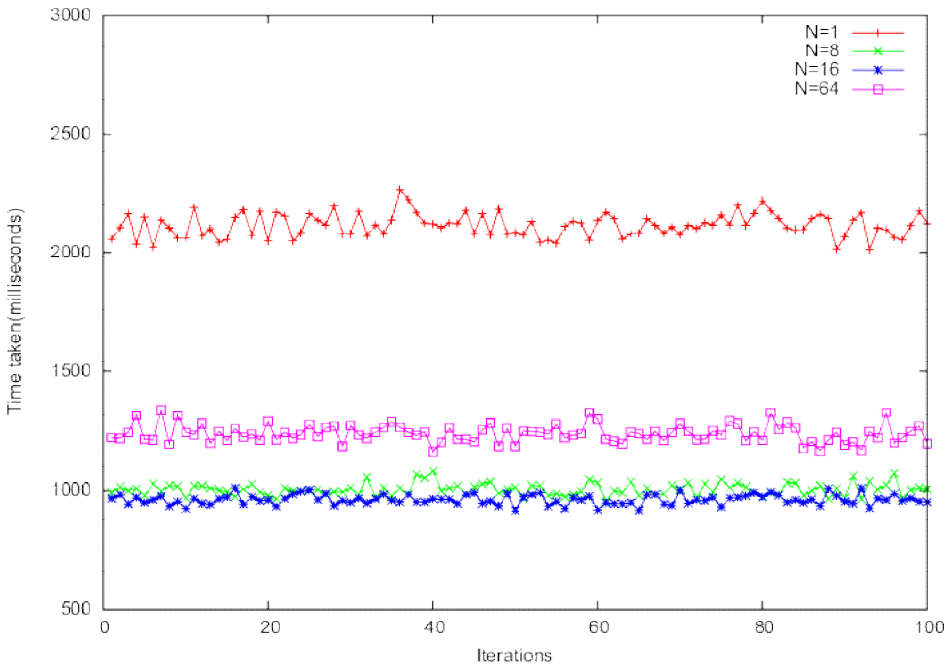
Results

- In cases 2-4, 4 execution sites created
  - Case 1: Without using the locality model.
  - Case 2: 1 thread
  - Case 3: 8 threads
  - Case 4: 439 threads



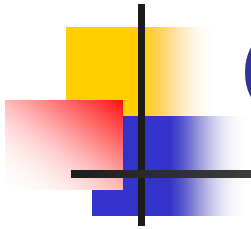
Results 2

- Execution sites dynamically created when load in the existing ES reaches N threads



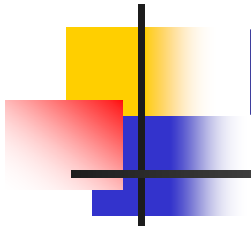
RTS York

THE UNIVERSITY of York



Conclusion

- Locality model has been proposed which
  - Groups together threads and objects in execution sites.
  - Execution sites can be allocated statically or dynamically.
  - System architecture is visible to the application.
- Experiments show considerable better performance.



Future Work

- Tests based on real-time applications.
- Constrains on higher priority real-time threads to execution sites.
- Locality for method area.
- Extensions
  - execution of multiple RTJVMs
  - heterogeneous systems