ET International

HPC Runtime Software

ET International
Rishi Khan
SC'11

Current Programming Models

- Shared Memory Multiprocessing
  - OpenMP – fork/join model
  - Pthreads – Arbitrary SMP parallelism (but hard to program/debug)
  - Cilk – Work Stealing (only good for recursive parallelism)
- Distributed Memory Multiprocessing
  - MPI – Bulk synchronous Parallelism
  - SHMEM, UPC - PGAS
- Hybrid Models
  - MPI + OpenMP (needed to get performance on multi-core, multi-node systems)
- Heterogeneous Accelerator Parallelism
  - CUDA, OpenCL

ET International

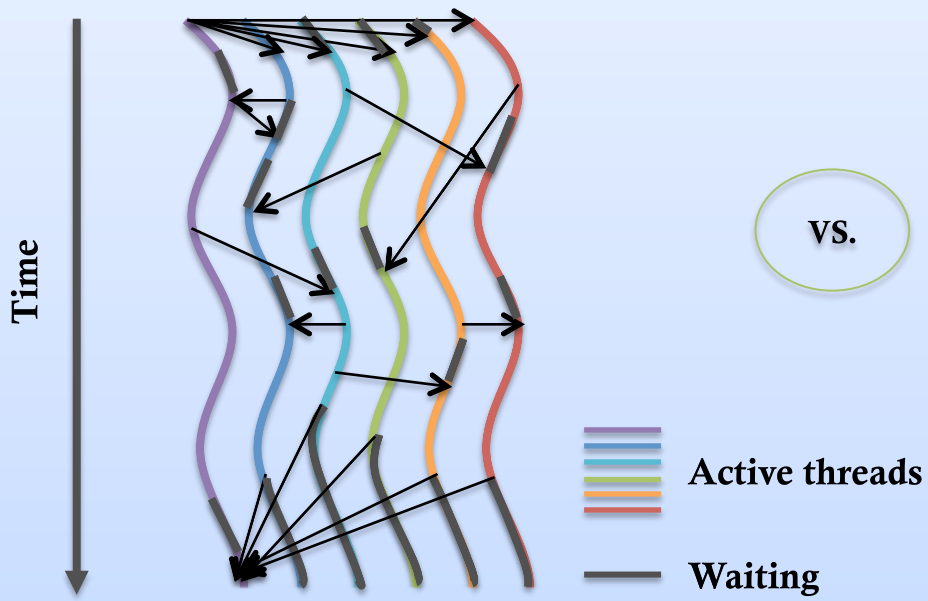
Problem Statement

ET International

1. Heterogeneous systems require multiple languages and programming models
   - e.g. MPI across nodes, OpenMP across cores, OpenCL across GPUs
2. Current programming models are based on the idea of ‘communicating sequential processes’ (CSP)
   - Difficult to program and debug.
   - Difficult to express dynamic parallelism
   - Does not take advantage of dynamic availability of resources
     - Extremely hard to exploit programs with irregular and/or global data accesses

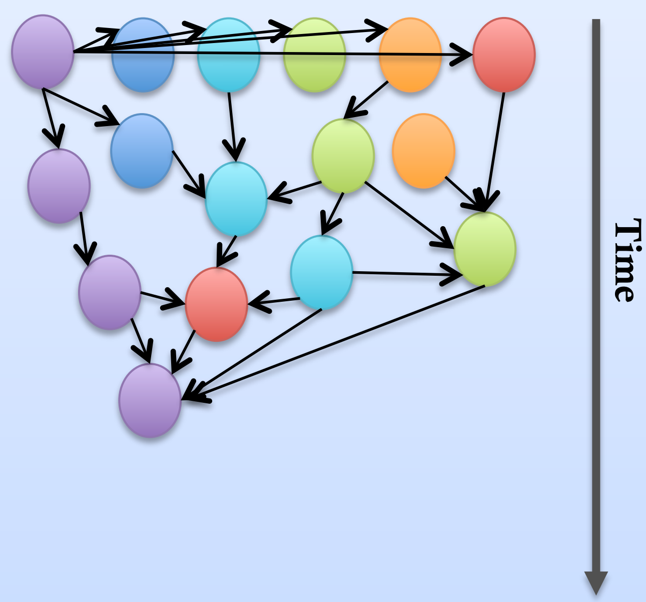
Runtime System Comparisons

MPI, OpenMP, OpenCL



vs.

New Runtime Systems



- Communicating Turing Machines
- Bulk Synchronous
- Message Passing

- Asynchronous Event-Driven Tasks
- Dependencies
- Constraints
- Resources
- Active Messages

ET International

Solution

- Express program as tasks with runtime dependencies and constraints
  - Data: input arguments
  - Control: must run before/after certain tasks
  - Resource: locks, CPU or GPU, etc
- Tasks can run to completion once all runtime dependencies and constraints are met
- Runtime system determines which tasks to run based on runtime resource availability.
- ETI implements this solution in a technology called "SWARM"

What is SWARM?

- SWift Adaptive Runtime Machine (SWARM):
  - Runtime system for heterogeneous large-scale systems
  - Implements an execution model based on specially tagged tasks:
    - Non-preemptible pieces of code.
    - Tagged with dependences, constraints, and resource demands.
    - Scheduled when all dependencies and constraints are satisfied.
    - Once scheduled, runs to completion in a non-blocking fashion.
    - These non-blocking tagged tasks are called codelets.
- Goal:
  - Unified runtime system for heterogeneous distributed parallel systems
  - Supplant and synergize the separate abilities of MPI, OpenMP, and OpenCL.

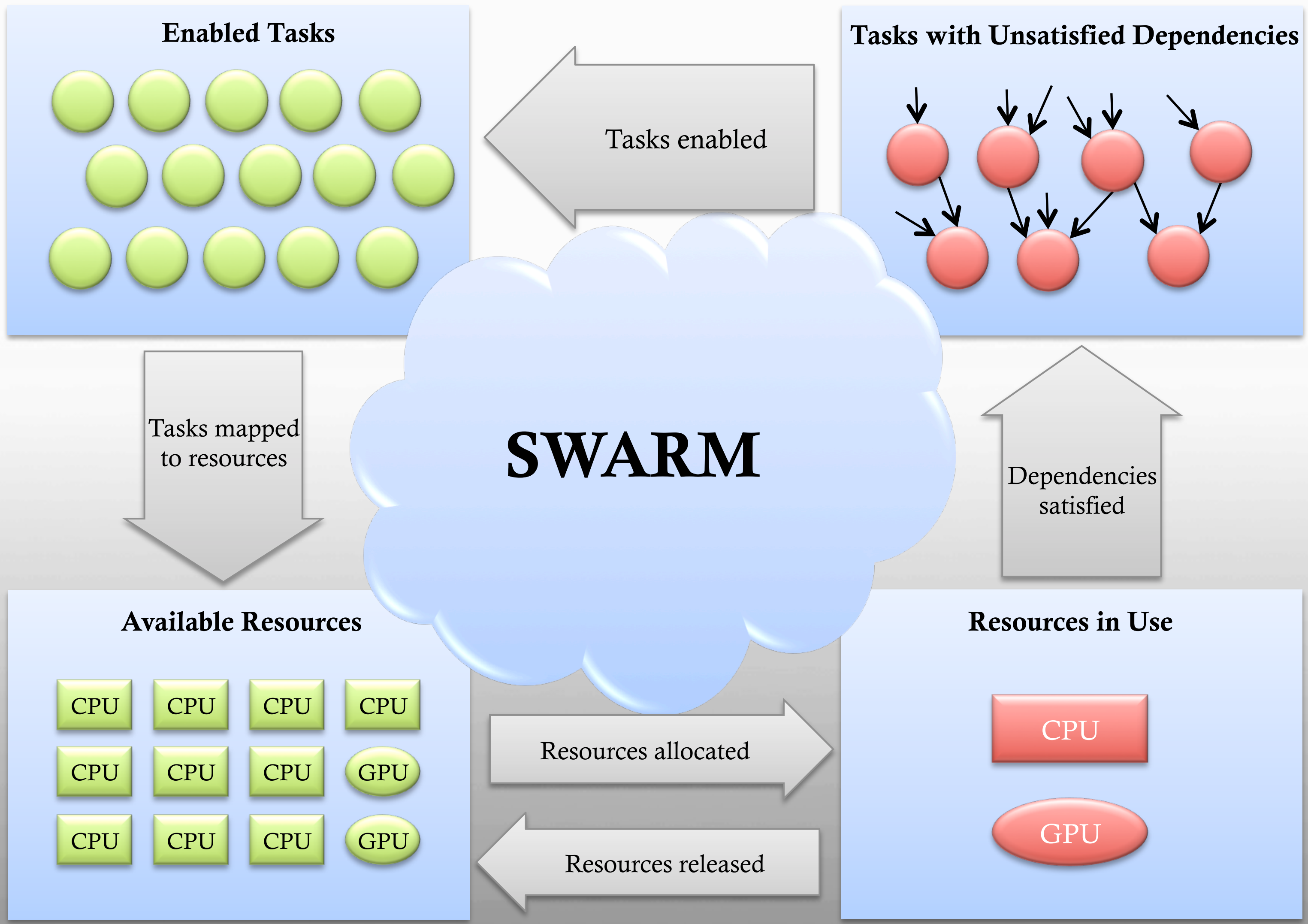
ET International

How does SWARM achieve its goals?

ET International

- **Two-level threading system**
  - First level are heavy-weight and bound to processing resource
  - Second level light-weight threads run non-preemptively
- **Object-oriented design** which is easily extended to new architectures and heterogeneous systems
  - Working across cores and nodes and heterogeneous devices
- All runtime resources are accessed through **split-phase non-blocking asynchronous operations**
  - The result of puts/gets are scheduled later using asynchronous callbacks
- Takes a **dynamic view of the computation and the machine**
  - in contrast to static mapping found in current programming models

SWARM Execution Overview



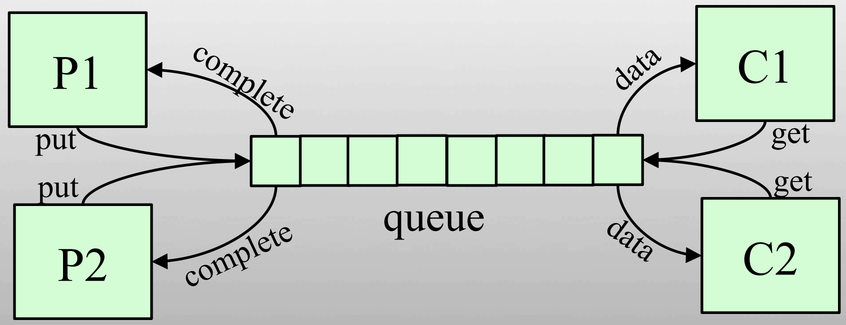
Runtime Resource Access

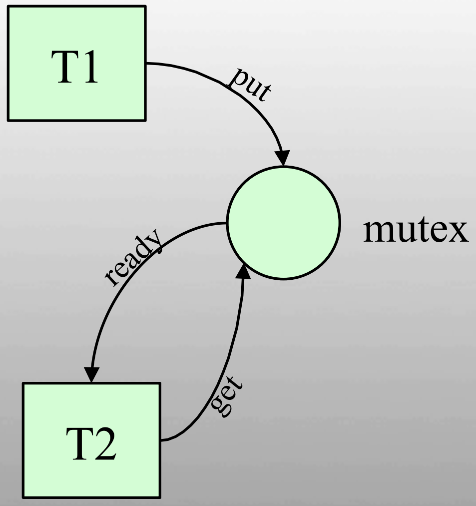
- All communication is through asynchronous split-phase transactions between resources, e.g.:
  - Async procedure call: put/get into procedure resources
  - Data storage: put/get to storage resource
- Two basic resource access patterns:
  - Producer passes key to consumer

  - Producer and consumer know resource key *a priori*

Motivation - Resource Sharing

- Allow for access to limited quantities of a resource
  - Example: mutex, queue
  - Producer “puts”, Consumer “gets”
  - Use a put callback to let producer know the operation completed
  - Use a get callback to let consumer know when the resource is available.





Key Features

- Exposing implicit parallelism
- Manage asynchrony
- Migration of data structures, work, global control
- Global namespace
- Hierarchy of locales for data locality and affinity
- Runtime Introspection
- Dynamic Adaptive Runtime System
- Solution to multicore/multinode problem that is user transparent to physical parallelism
- Diversity of scheduling domains & policies of tasks and resources
- Readable intermediate representation

ET International

Moving Global Control

```
while (visited_list)
{
  foreach(v in visited_list)
    foreach(n in neighbors(v))
    {
       if (!visited(n))
       {
         new_visited_list[pos++] = n;
         parent[n] = v;
       }
    }
  }
  swap_lists(new_visited_list, visited_list);
}
```

Run this at the owner of n. (the `if (!visited(n)) { … }` block)

SWARM: Key Concepts

- SWift Adaptive Runtime Machine:
  - Unifies across nodes, cores, and accelerators
  - Dynamically maps applications needs to available resources
  - Provides expression of asynchronous programs to maximize performance and hide latency
  - Communication and synchronization is implicit in the task dependencies

ET International

SWARM Availability Presently

- Current features
  - HAL backends for x86 (32 and 64-bit), POSIX
  - Scheduling of codelets
  - Create dependencies between codelets
  - Basic network support via TCP/IP
  - SCALE Codelet IR Language
  - API Documentation and Programmers Guide
- Early Access Release SWARM 0.7.0 available now:
  - http://www.etinternational.com/swarm
- New version by early December
  - Full locale support (scheduling and memory)
  - Full abstraction of hardware/OS in HAL
  - Proper network stack
  - Codelet/function symmetry

ET International

SWARM Future Plans

- Hardware Support
  - Intel MIC, Runnemeade
  - GPU, Adapteva
- Legacy Support
  - Work with MPI/OpenMP and other runtimes
    - Via recompilation (e.g. OpenMP)
    - Operate side-by-side (e.g. MPI)
    - Via DLL injection (e.g. OpenCL)
  - UPC, SHMEM support
- Language
  - Wrestling with higher level language
    - Detailed language for experts, yet simple for Joe programmer
- Monitoring and Debugging

ET International

ET International

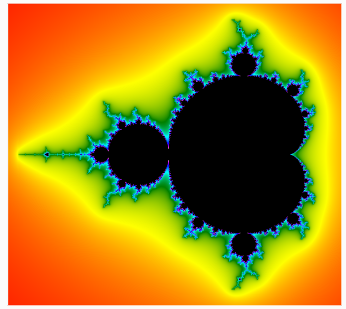
Key Takeaways

- Contexts Where SWARM helps
  - Irregular loads
  - Long latency operations
  - Resource constraints other than CPU
  - Heterogeneous systems
- Benefits
  - Programming Productivity
  - Deliver higher throughput and higher performance
  - Power efficiency
  - Purchase flexibility
- Key Runtime Concepts
  - Asynchronous Split-Phase Resource Access
  - Hierarchical Event Driven Scheduling
  - Abstraction of resources for unified heterogeneous access
- Experiences
  - SWARM Runtime system
  - SWARM SCALE Codelet IR Langauge

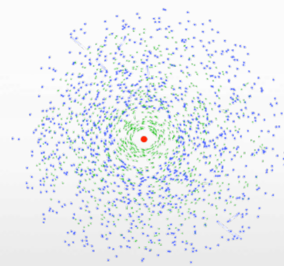
ET International

Case Studies

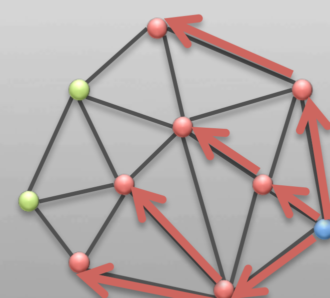
- Mandelbrot



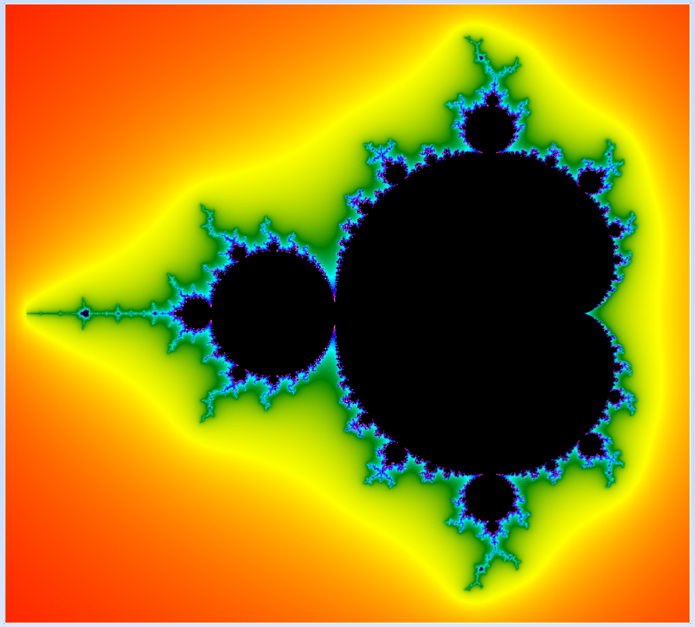
- Barnes-hut N-body problem

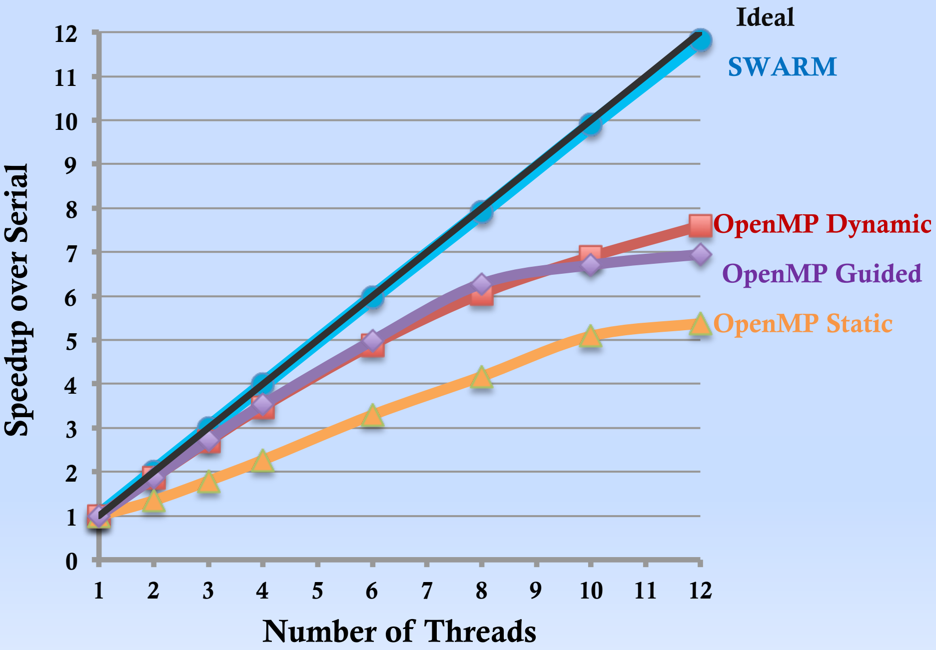


- Graph500



Mandelbrot

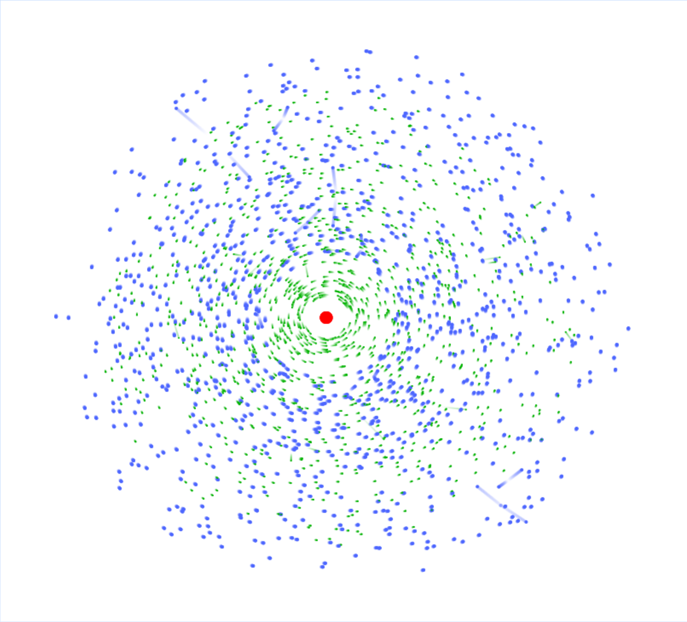


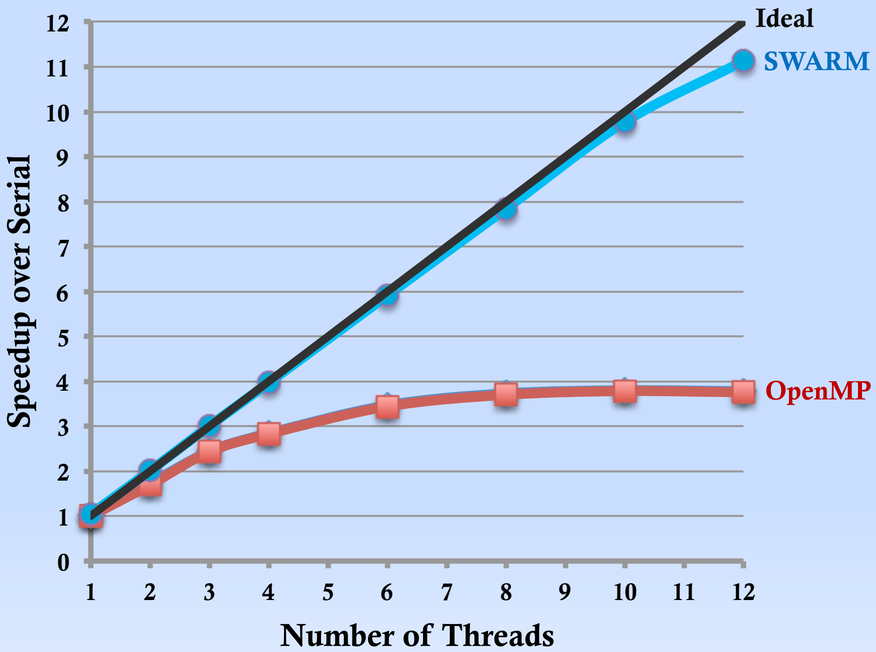


Mandelbrot

ET International

Barnes-Hut





Barnes-Hut

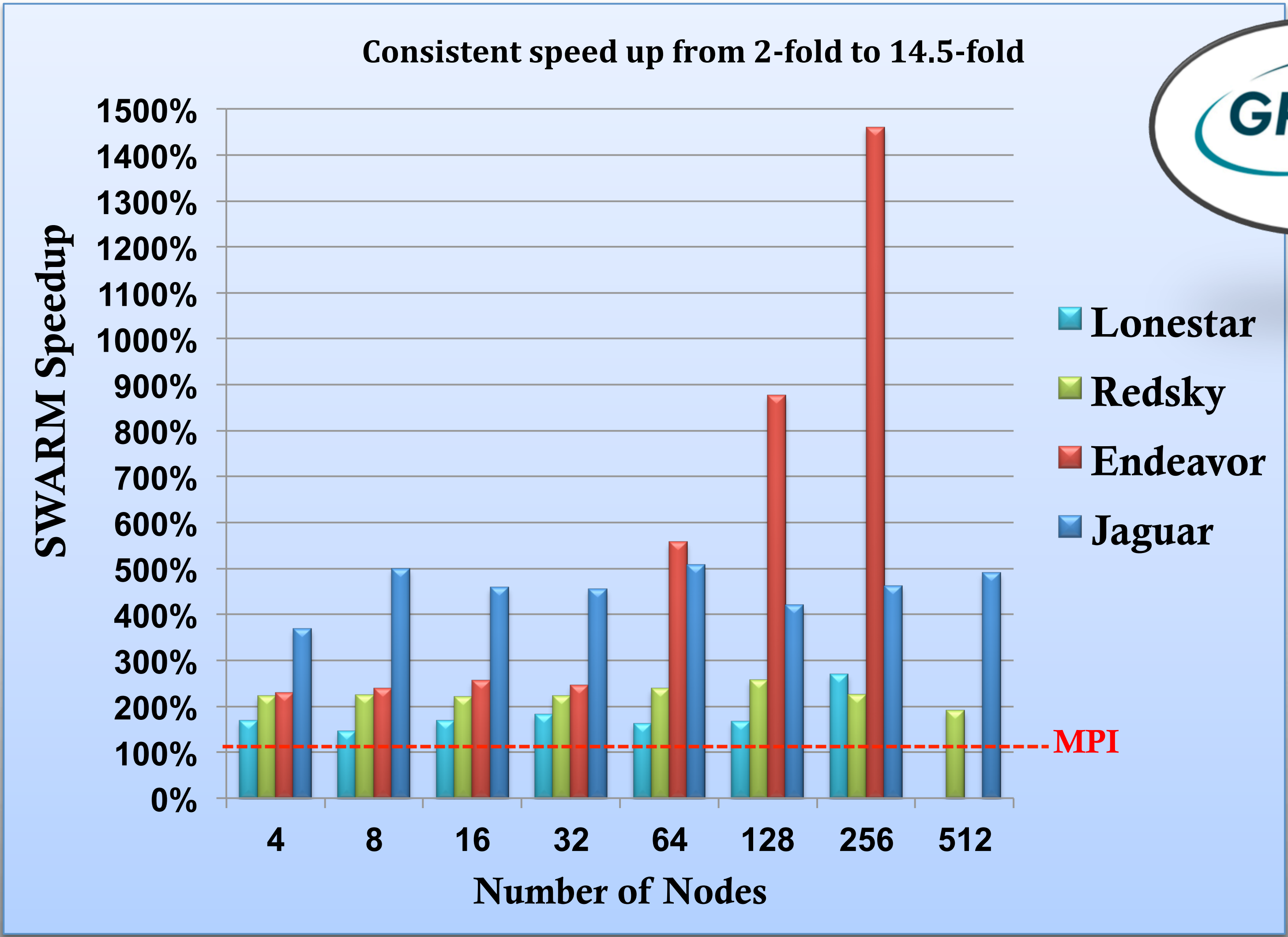
Graph 500 Implementation with SWARM

- Graph500: New supercomputing benchmark for more realistic application workloads
- Ported to SWARM and produced results on 4 different supercomputers.

Supercomputer Name	Sandia Redsky	TACC Lonestar	Intel Endeavor	ORNL Jaguar
Processor Type	Nehalem X5570	Westmere 5680	Westmere X5670	Cray XT5-HE
Processor Speed	2.93 GHz	3.33 GHz	2.93 GHz	2.6 GHz
Processors per Node	8	12	12	12
Main memory size	12GB/node	24GB/node	24GB/node	16GB/Node

ET International

SWARM/MPI Performance Comparison



ET International

Advantages of SWARM on GRAPH 500

- Lower type overhead
- Active messages - fewer copies and round trips
- Share address space on same node
- Monitor and allocate cache utilization
- Idle threads can steal work from other threads
- Effective substitute for MPI + OpenMP + Active Messages – All in one package with lower overheads

ET International

SWARM: Key Concepts

- SWift Adaptive Runtime Machine:
  - Unifies across nodes, cores, and accelerators
  - Dynamically maps applications needs to available resources
  - Provides expression of asynchronous programs to maximize performance and hide latency
  - Communication and synchronization is implicit in the task dependencies

ET International

SWARM Availability Presently

- Current features
  - HAL backends for x86 (32 and 64-bit), POSIX
  - Scheduling of codelets
  - Create dependencies between codelets
  - Basic network support via TCP/IP
  - SCALE Codelet IR Language
  - API Documentation and Programmers Guide
- Early Access Release SWARM 0.7.0 available now:
  - http://www.etinternational.com/swarm
- New version by early December
  - Full locale support (scheduling and memory)
  - Full abstraction of hardware/OS in HAL
  - Proper network stack
  - Codelet/function symmetry

SWARM Future Plans

- Hardware Support
  - Intel MIC, Runnemeade
  - GPU, Adapteva
- Legacy Support
  - Work with MPI/OpenMP and other runtimes
    - Via recompilation (e.g. OpenMP)
    - Operate side-by-side (e.g. MPI)
    - Via DLL injection (e.g. OpenCL)
  - UPC, SHMEM support
- Language
  - Wrestling with higher level language
    - Detailed language for experts, yet simple for Joe programmer
- Monitoring and Debugging

ET International

ET International

Key Takeaways

- Contexts Where SWARM helps
  - Irregular loads
  - Long latency operations
  - Resource constraints other than CPU
  - Heterogeneous systems
- Benefits
  - Programming Productivity
  - Deliver higher throughput and higher performance
  - Power efficiency
  - Purchase flexibility
- Key Runtime Concepts
  - Asynchronous Split-Phase Resource Access
  - Hierarchical Event Driven Scheduling
  - Abstraction of resources for unified heterogeneous access
- Experiences
  - SWARM Runtime system
  - SWARM SCALE Codelet IR Langauge