Three Challenges and Three Solutions for Exascale Computing

Amir Kamil and Katherine Yelick
UC Berkeley/LBNL
June 28, 2012
Three Challenges and Solutions

❖ Three challenges
  ▪ Hierarchy
  ▪ Heterogeneity
  ▪ Resilience

❖ Three solutions
  ▪ Programming models
  ▪ Compiler/runtime infrastructure
  ▪ Domain-specific libraries, languages, and specializers
Hierarchy Challenge

- Locality essential to achieving good performance at large scale, minimizing energy use

- Parallel locality no longer a binary value (local vs. not local)
  - Non-uniform memory access (NUMA) within single shared-memory node
  - Manycore systems may have multiple levels of NUMA-ness
Heterogeneity Challenge

- Machines composed of heterogeneous components due to performance vs. energy tradeoffs

- Components may have different capabilities (e.g. accelerators) or same capabilities at different performance and energy points
  - 3 of the top 5 machines in 11/11 include GPUs

Cell phone processor
(0.1 Watt, 4 Gflop/s)

Server processor
(100 Watts, 50 Gflop/s)
Increasing scale means increasing probability of component failure during a program run.

Tradeoff between power consumption and error rates at the chip level increases likelihood of undetected errors.
Three Solutions

- Expose problems to user in the programming model
  - Force each user to deal with the problems themselves

- Generic solutions in the compiler, runtime, and hardware

- Domain-specific solutions in libraries, languages, and specializers
  - Expert programmers solve the problems
Hierarchical Programming Models

- Programming model solutions for hierarchy
  - Hierarchical partitioned global address space (HPGAS) memory model
  - Recursive single-program, multiple data (RSPMD) execution model

Diagram:
- Span 1: (core local)
- Span 2: (processor local)
- Span 3: (node local)
- Span 4: (global)

Program Start: 0, 1, 2, 3, 4, 5, 6, 7

0, 1, 2, 3
0, 1
2, 3
4, 5
4, 5, 6, 7
6, 7

0, 1, 2, 3, 4, 5, 6, 7

Barriers: 0, 1, 2, 3, 4, 5, 6, 7
Exchange: 0, 1, 2, 3, 4, 5, 6, 7
Partitioned Global Address Space

- Partitioned global address space (PGAS) abstraction provides illusion of shared memory on non-shared memory machines

- Pointers can reference local or remote data
  - Location of data can be reflected in type system
  - Runtime handles any required communication

```java
double[1d] local srcl = new double[0:N-1];
double[1d] srcg = broadcast srcl from 0;
```
Hierarchical PGAS

- PGAS model can be extended to hierarchical arrangement of memory spaces
- Pointers have varying span specifying how far away the referenced object can be
  - Reflect communication costs
- Pointer span can be inferred by compiler through hierarchical pointer analysis
Single Program, Multiple Data

- Single program, multiple data (SPMD): fixed set of threads execute the same program image

```java
public static void main(String[] args) {
    System.out.println("Hello from thread "+ Ti.thisProc());
    Ti.barrier();
    if (Ti.thisProc() == 0)
        System.out.println("Done.");
}
```

Program Start

- Print
- Print
- Print
- Print
- Print
- Print
- Print
- Print
- Print

Barrier

Program End

Print
Recursive SPMD

- Threads recursively subdivided into smaller teams in tree-like structure
  - Allow arbitrary hierarchies (e.g. unbalanced trees)
  - Can match machine structure
New language constructs for assigning tasks or data to teams

- Lexical scope prevents some types of deadlock, dynamic checks prevent others

```c
teamsplit(T) {
    Ti.barrier();
    teamsplit(T2) {
        Ti.barrier();
    }
}

arr.exchange(data);
```
RSPMD allows easy expression of hierarchical algorithms

- Inherent hierarchy such as divide and conquer
- Optimizations for hierarchical machines

Optimized Distributed Sort (Cray XT4)
(10,000,000 elements/core, 10,000 samples/core)
Programming Model
Disadvantages

- Forces user to manually perform hierarchical locality optimizations
  - May be machine-specific and not generalize to arbitrary machine structures

- Algorithmic decomposition may not match machine structure
  - Matrix-vector multiplication with 2D decomposition requires row and column teams
    - Both can’t map to low levels in machine hierarchy
Other Approaches to Hierarchy

- **Compiler/runtime ideas**
  - Pointer analysis for HPGAS can be extended to RSPMMD as well
  - Can use analysis to estimate communication costs between threads and map threads to machine
  - Dynamic analysis can increase precision at low cost

- **Domain-specific ideas**
  - Common communication patterns can be optimized in a library or specializer
    - Dense grid applications generally use static ghost zones, resulting in regular, repeated communication
Summary

- Hierarchical programming models simplify expression of hierarchical algorithms and enable good performance on hierarchical machines.
- Need to be combined with compiler/runtime and domain-specific solutions to further improve productivity and performance.
- We believe that all three challenges (hierarchy, heterogeneity, fault-tolerance) require a combination of programming model, compiler/runtime, and domain-specific solutions.