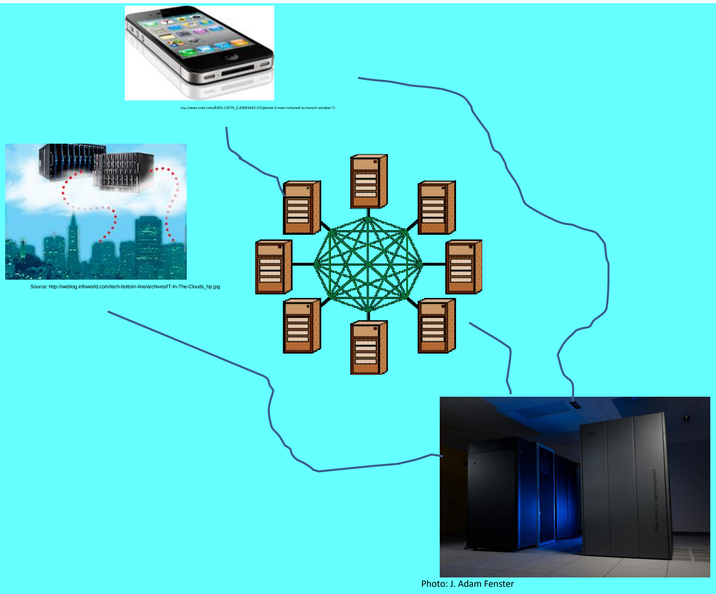
Collaborative Research:XPS:CLCCA:
Performance Portable Abstractions for Large-Scale Irregular Computations

Sandhya Dwarkadas and Michael L. Scott
University of Rochester

Srinivas Aluru
Georgia Institute of Technology

Background and Overview

- Computations on extremely large datasets are increasingly important to progress in a wide variety of commercial and scientific domains
- Systems such as MapReduce and GraphLab make the ability to harness distributed compute power more readily available
- Scalable parallelism requires runtime knowledge of application data structures and access patterns
- Our goal is to develop performance portable abstractions that allow the runtime system to infer application access patterns and localities



Research Highlights

- Deterministic Parallel Ruby provides scalable parallelism for scripting languages
- Sharing-Aware Mapping provides users and application developers the ability to harness scalable parallelism in a topology-oblivious manner

Locality-Aware API

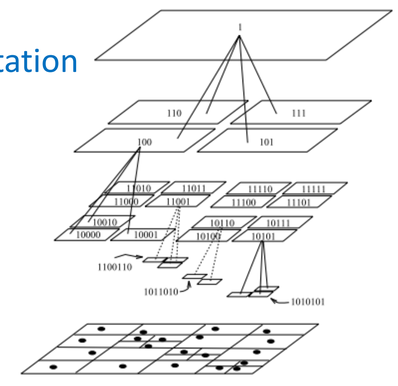
- Algorithmic abstractions that capture access locality
- Initial target application subclass: tree-based data representation
  - Computational biology
  - Genome assembly
  - Molecular dynamics

Proposed API Extensions for Trees

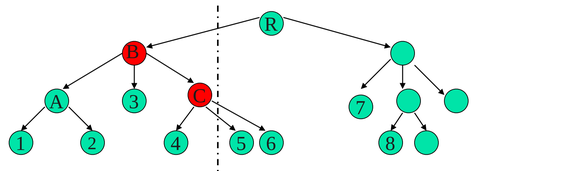
$$\texttt{generate}(u) \longrightarrow \langle \mathrm{CS}(u), \texttt{DEPENDENCY} \rangle \qquad (1)$$

$$\texttt{combine}(u, v) \longrightarrow u', \quad v \in \mathrm{CS}(u) \qquad (2)$$

Example: Quad-Tree Representation



- Runtime is responsible for the distributed representation and storage of the tree



Local tree of P1 stored as postorder array

Local tree of P2 stored as postorder array

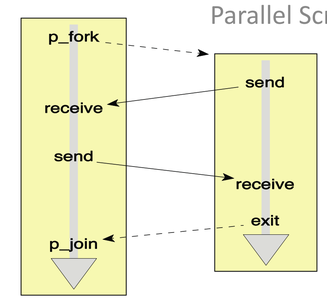
P1: 1 2 A 3 4 C B P2: 5 6 C 7 8 ……R

Out of order nodes

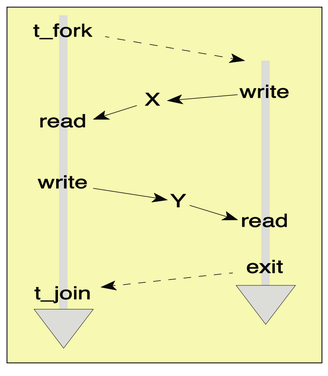
Deterministic Parallel Ruby

- Split-merge parallelism
- Automatic checking of branch independence
- TARDIS race detector
- Language-level (semantic) conflict detection

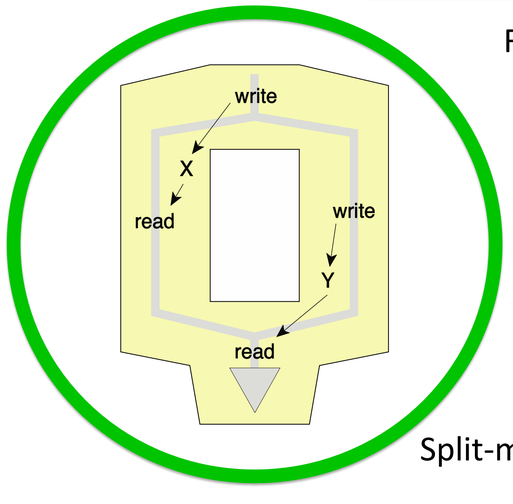
Dynamic Enforcement of Determinism in a Parallel Scripting Language [PLDI 2014]



Multi-process



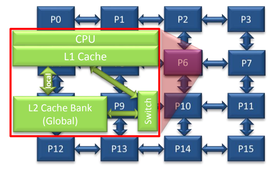
Fork-join



Split-merge

Sharing-Aware Mapping (SAM)

- Reality: Programmers and users must be aware of hardware topology and non-uniform sharing



* Photo courtesy crunchgear.com

Need better support to improve programmer and user productivity

- SAM uses low-cost hardware performance counters commonly available on modern processors to identify and separate data and resource sharing
- Results demonstrate that adaptive online sharing-aware mapping at the scheduler level effectively localizes traffic due to sharing and minimizes resource contention for improved fairness

Data Sharing or Resource Contention: Toward Performance Transparency on Multicore Systems [Usenix ATC 2015]

Average speedup = 1.23
Min speedup = 0.96
Max speedup = 1.72