Microsoft Research

Faculty Summit

2014 15TH ANNUAL

Microsoft Research
Faculty Summit
2014 15TH ANNUAL

Parallel Programming in the Age of Ubiquitous Parallelism

Andrew Lenharth
Slides: Keshav Pingali
The University of Texas at Austin

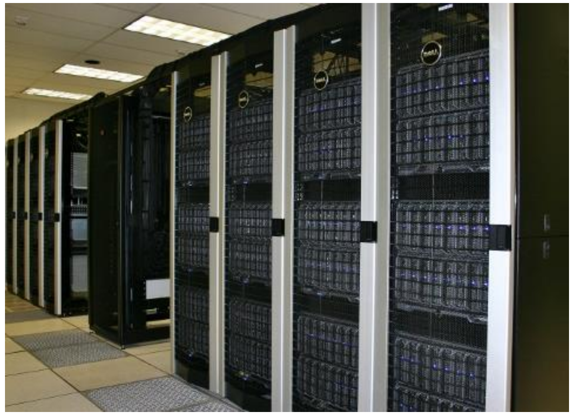
TEXAS

Microsoft

Parallel Programming in the Age of Ubiquitous Parallelism

Andrew Lenharth
Slides: Keshav Pingali
The University of Texas at Austin

Parallelism is everywhere



Texas Advanced
Computing Center

Laptops

Cell-phones

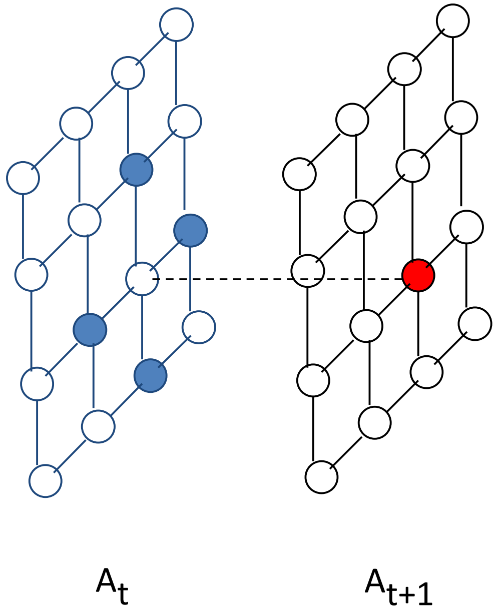
Parallel programming?

- 40-50 years of work on parallel programming in HPC domain
- Focused mostly on “regular” dense matrix/vector algorithms
  - Stencil computations, FFT, etc.
  - Mature theory and tools
- Not useful for “irregular” algorithms that use graphs, sets, and other complex data structures
  - Most algorithms are irregular ☹
- Galois project:
  - New data-centric abstractions for parallelism and locality
  - Galois system for multicores and GPUs

“The Alchemist”
Cornelius Bega (1663)

HPC example

- Finite-difference computation
- Algorithm
  - Operator: five-point stencil
  - Different schedules have different locality
- Regular application
  - Application can be parallelized at compile-time



Jacobi iteration, 5-point stencil

```
//Jacobi iteration with 5-point stencil
//initialize array A
for time = 1, nsteps
   for <i,j> in [2,n-1]x[2,n-1]
       temp(i,j)=0.25*(A(i-1,j)+A(i+1,j)+A(i,j-1)+A(i,j+1))
   for <i,j> in [2,n-1]x[2,n-1]:
       A(i,j) = temp(i,j)
```

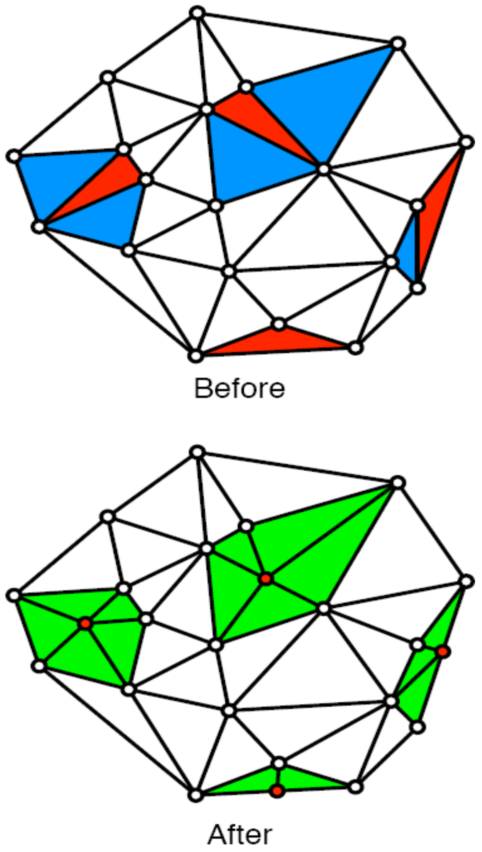
Irregular example

```
Mesh m = /* read in mesh */
WorkList wl;
wl.add(m.badTriangles());
while (true) {
   if (wl.empty()) break;
   Element e = wl.get();
   if (e no longer in mesh)
      continue;
   Cavity c = new          Cavity(e);
   c.expand();
   c.retriangulate();
   m.update(c);//update mesh
   wl.add(c.badTriangles());
}
```

Computation-centric view of parallelism

- Where is parallelism in program?
  - Loop: do static analysis to find dependence graph
- Static analysis fails to find parallelism.
  - May be there is no parallelism in program?

Data-centric view of algorithm

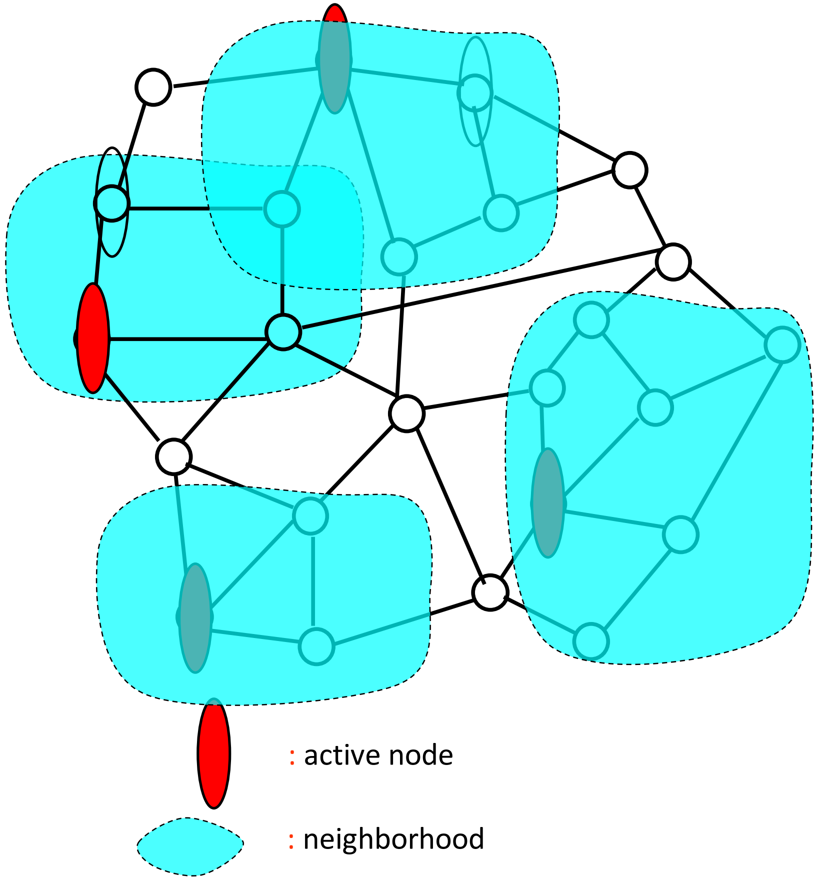


Delaunay mesh refinement (DMR)
Red Triangle: badly shaped triangle
Blue triangles: cavity of bad triangle

- Algorithm
  - composition of atomic actions on data structures
- Actions: operator
  - DMR: {find cavity, retriangulate, update mesh}
- Composition of actions:
  - specified by a schedule
- Parallelism
  - disjoint actions can be performed in parallel
- Parallel data structures
  - graph
  - worklist of bad triangles

Operator formulation of algorithms

- Active element
  - Site where computation is needed
- Operator
  - Computation at active element
  - Activity: application of operator to active element
- Neighborhood
  - Set of nodes/edges read/written by activity
  - Distinct usually from neighbors in graph
- Ordering : scheduling constraints on execution order of activities
  - Unordered algorithms: no semantic constraints but performance may depend on schedule
  - Ordered algorithms: problem-dependent order
- Amorphous data-parallelism
  - Multiple active nodes can be processed in parallel subject to neighborhood and ordering constraints

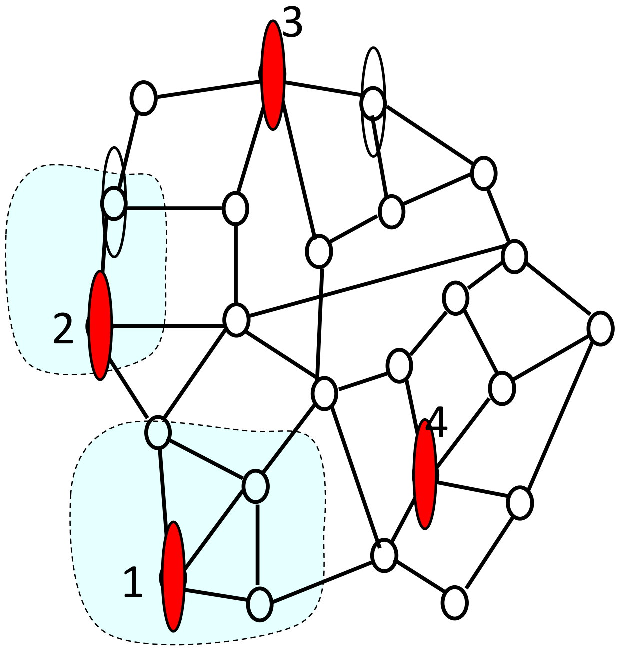


Parallel program = Operator + Schedule + Parallel data structure

Parallelization strategies: Binding Time

When do you know the active nodes and neighborhoods?

Compile-time	**Static parallelization (stencil codes, FFT, dense linear algebra)**
After input is given	**Inspector-executor (Bellman-Ford)**
During program execution	**Interference graph (DMR, chaotic SSSP)**
After program is finished	**Optimistic Parallelization (Time-warp)**



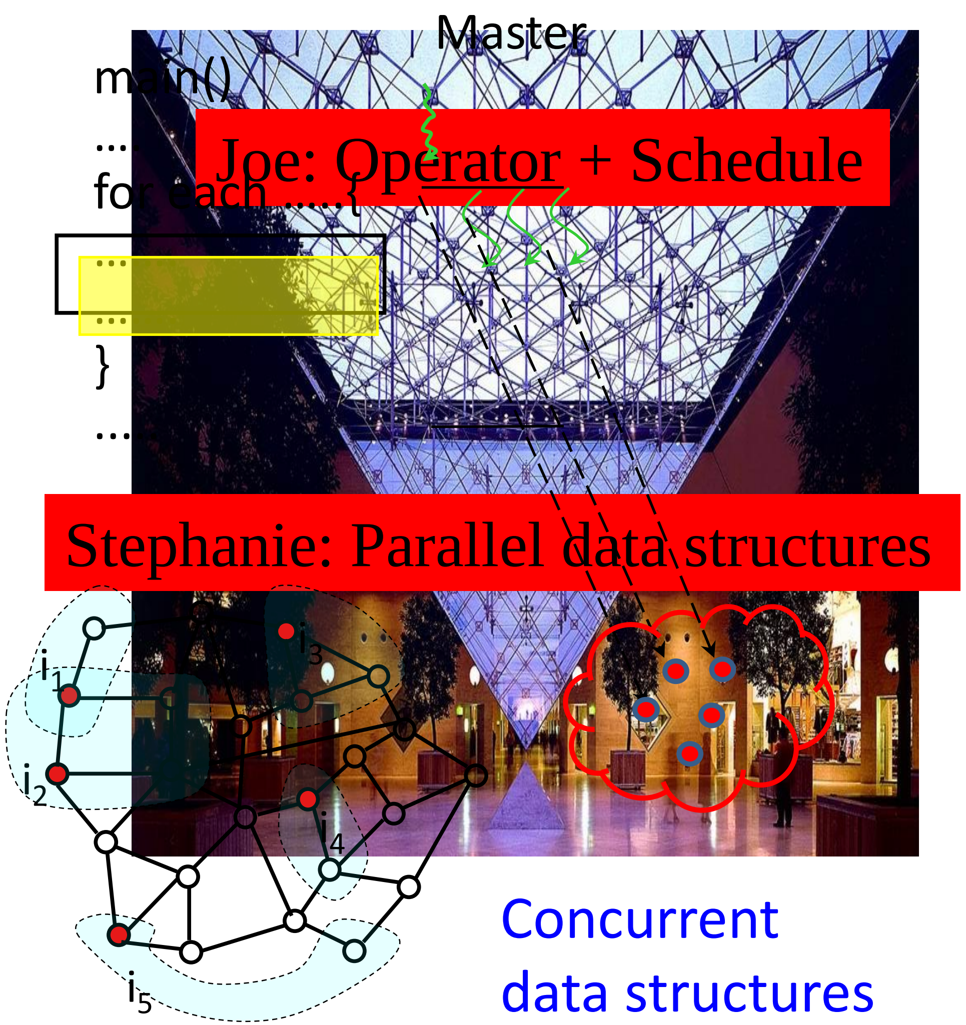
“The TAO of parallelism in algorithms” Pingali et al, PLDI 2011

Galois system

Parallel program = Operator + Schedule + Parallel data structures

- Ubiquitous parallelism:
  - small number of expert programmers (Stephanies) must support large number of application programmers (Joes)
  - cf. SQL
- Galois system:
  - Stephanie: library of concurrent data structures and runtime system
  - Joe: application code in sequential C++
    - Galois set iterator for highlighting opportunities for exploiting ADP

Joe Program



Concurrent data structures

“Hello graph” Galois Program

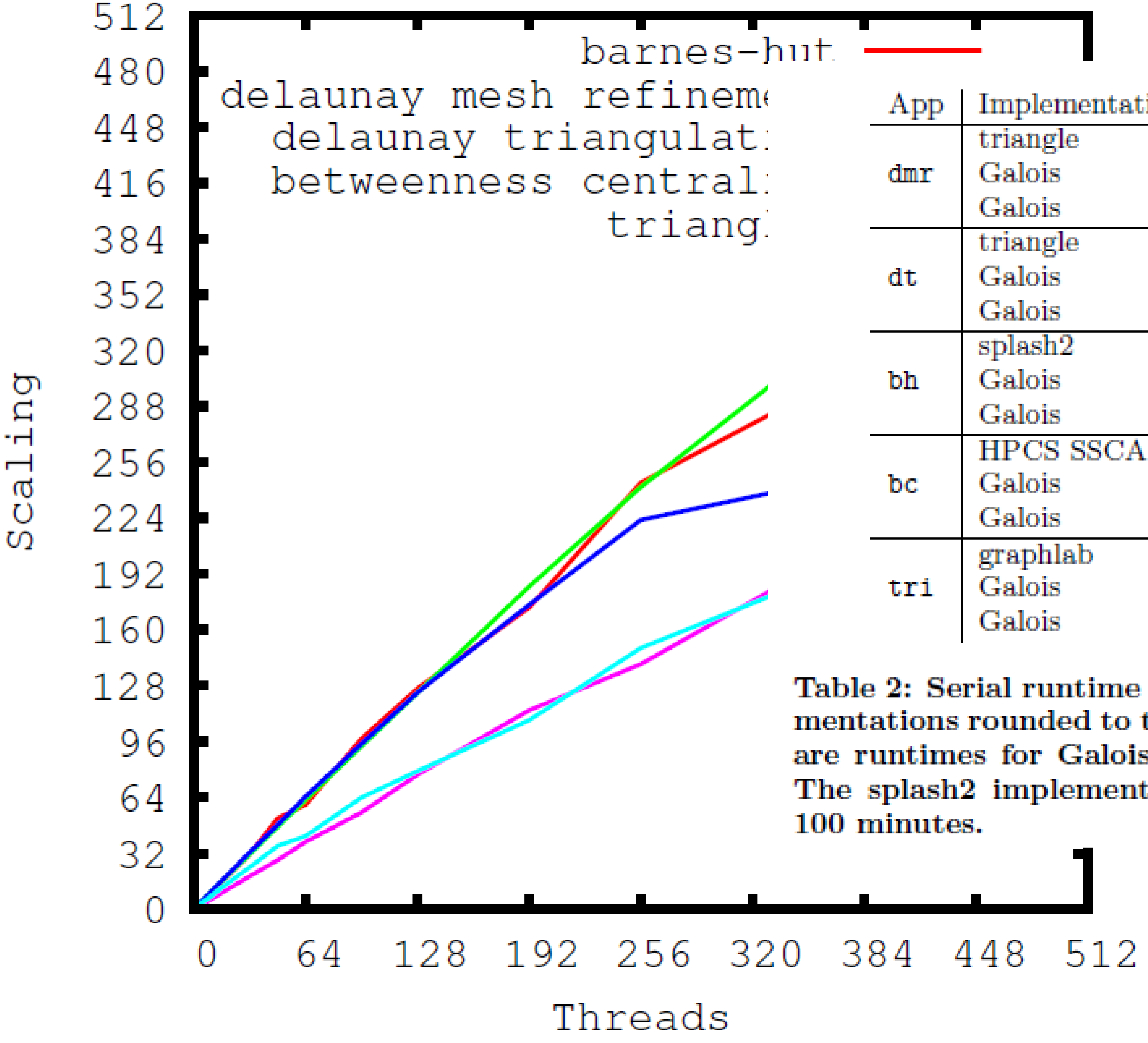
```
#include “Galois/Galois.h”
#include “Galois/Graphs/LCGraph.h”

struct Data { int value; float f; };

typedef Galois::Graph::LC_CSR_Graph<Data,void> Graph;
typedef Galois::Graph::GraphNode Node;

Graph graph;

struct P {
  void operator()(Node n, Galois::UserContext<Node>& ctx) {
    graph.getData(n).value += 1;
  }
};

int main(int argc, char** argv) {
  graph.structureFromGraph(argv[1]);
  Galois::for_each(graph.begin(), graph.end(), P());
  return 0;
}
```

Data structure Declarations

Operator

Galois Iterator

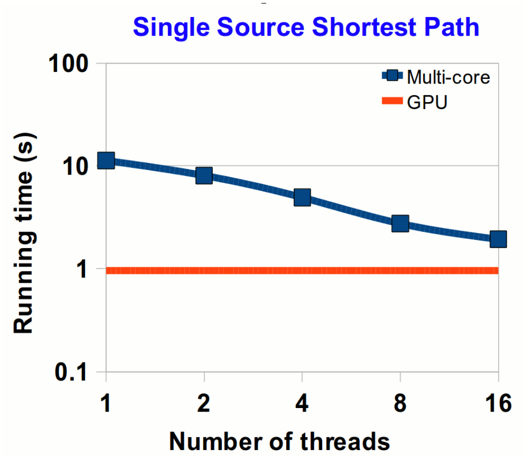
Galois: Performance on SGI Ultraviolet

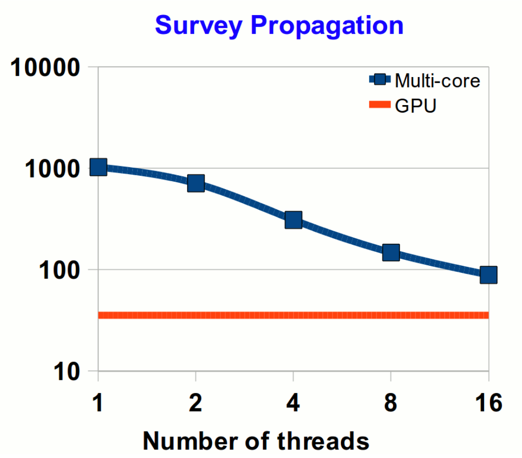


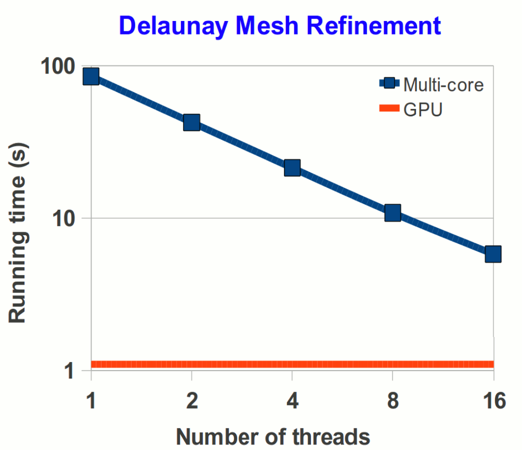
App	Implementation	Threads	Time (s)
dmr	triangle	1	96
	Galois	1	155.7
	Galois	512	0.37
dt	triangle	1	1185
	Galois	1	56.6
	Galois	512	0.18
bh	splash2	1	>6000
	Galois	1	1386
	Galois	512	3.55
bc	HPCS SSCA	1	6720
	Galois	1	5394
	Galois	512	21.6
tri	graphlab	2	531
	Galois	1	7.03
	Galois	512	0.028

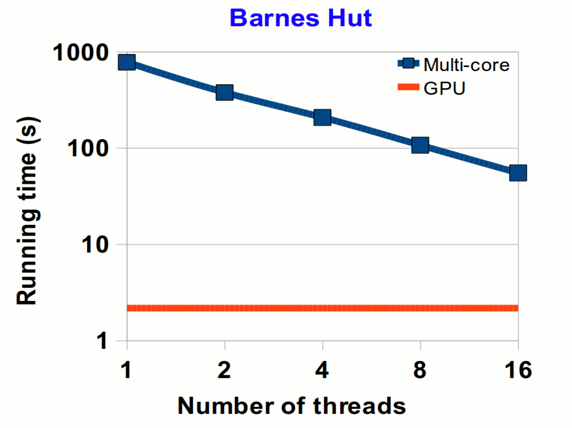
Table 2: Serial runtime comparisons to other implementations rounded to the nearest second. Included are runtimes for Galois algorithms at 512 threads. The splash2 implementation of bh timed out after 100 minutes.

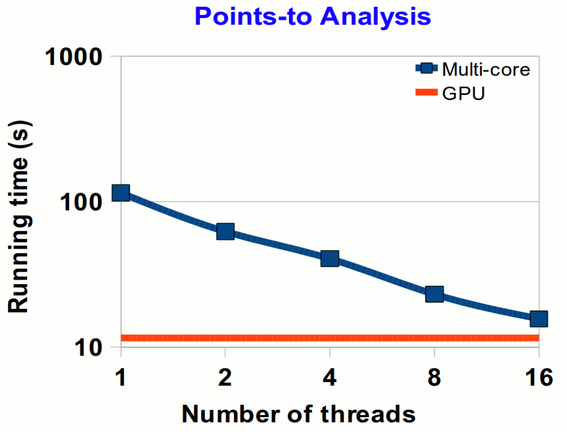
GPU implementation







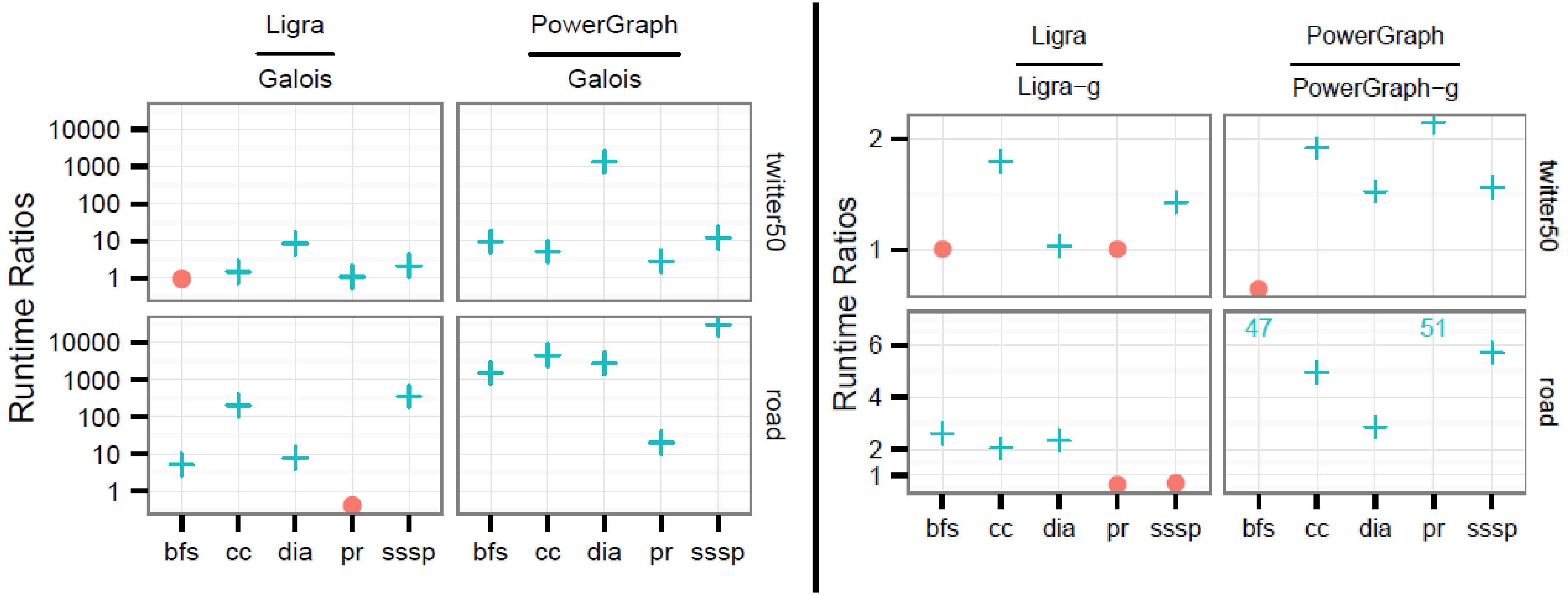




Multicore: 24 core Xeon
GPU: NVIDIA Tesla

Inputs:	SSSP: 23M nodes, 57M edges	SP: 1M literals, 4.2M clauses	DMR: 10M triangles
	BH: 5M stars	PTA: 1.5M variables, 0.4M constraints	

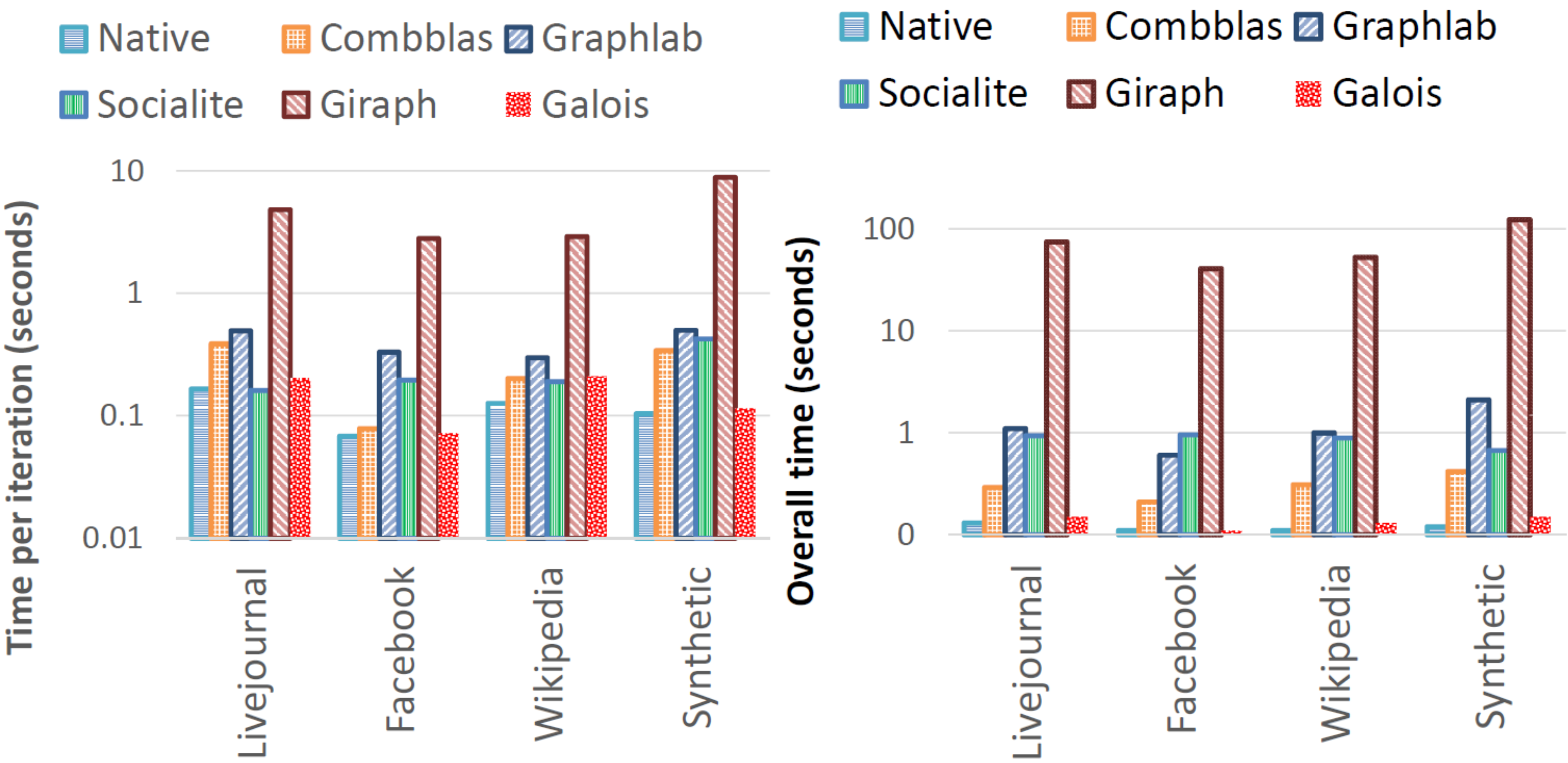
Galois: Graph analytics



- Galois lets you code more effective algorithms for graph analytics than DSLs like PowerGraph (left figure)
- Easy to implement APIs for graph DSLs on top on Galois and exploit better infrastructure (few hundred lines of code for PowerGraph and Ligra) (right figure)

“A lightweight infrastructure for graph analytics” Nguyen, Lenharth, Pingali (SOSP 2013)

Intel Study: Galois vs. Graph Frameworks

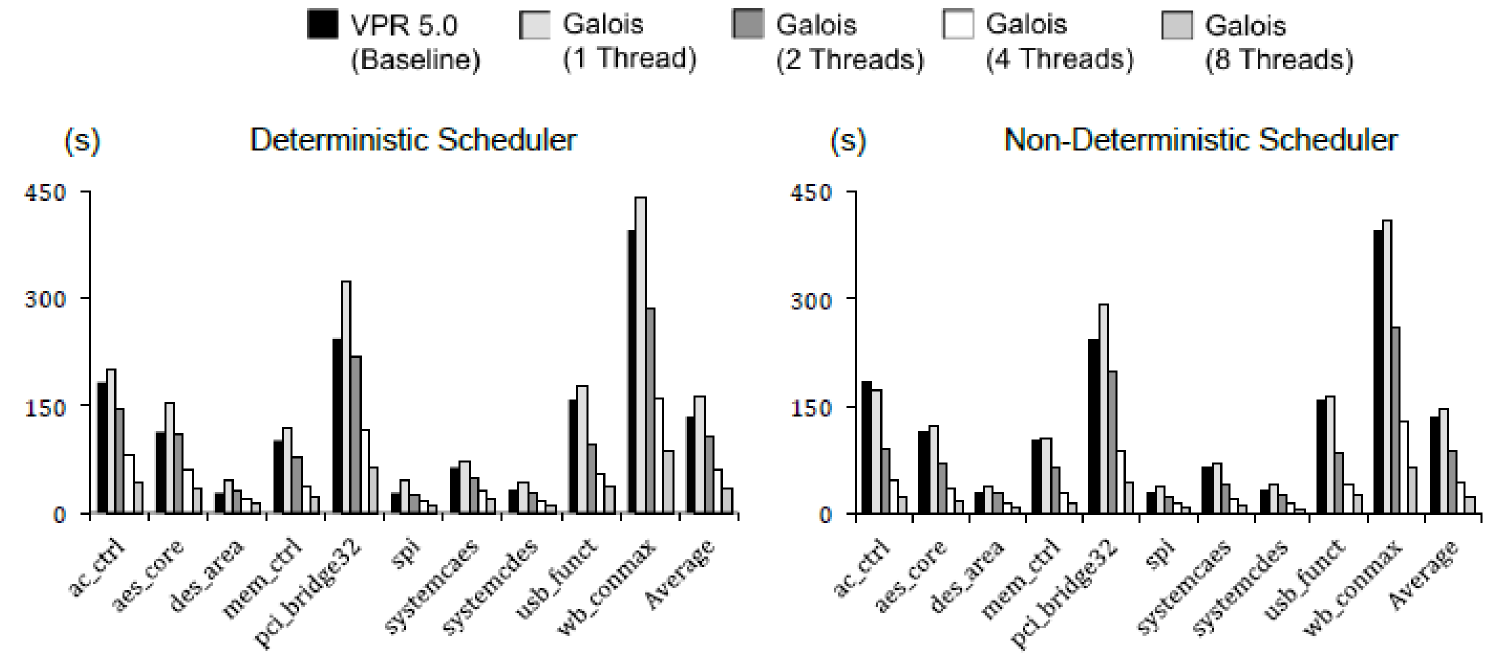


(a) PageRank (b) Breadth-First Search

“Navigating the maze of graph analytics frameworks” Nadathur et al SIGMOD 2014

FPGA Tools

Maze Router Execution Time

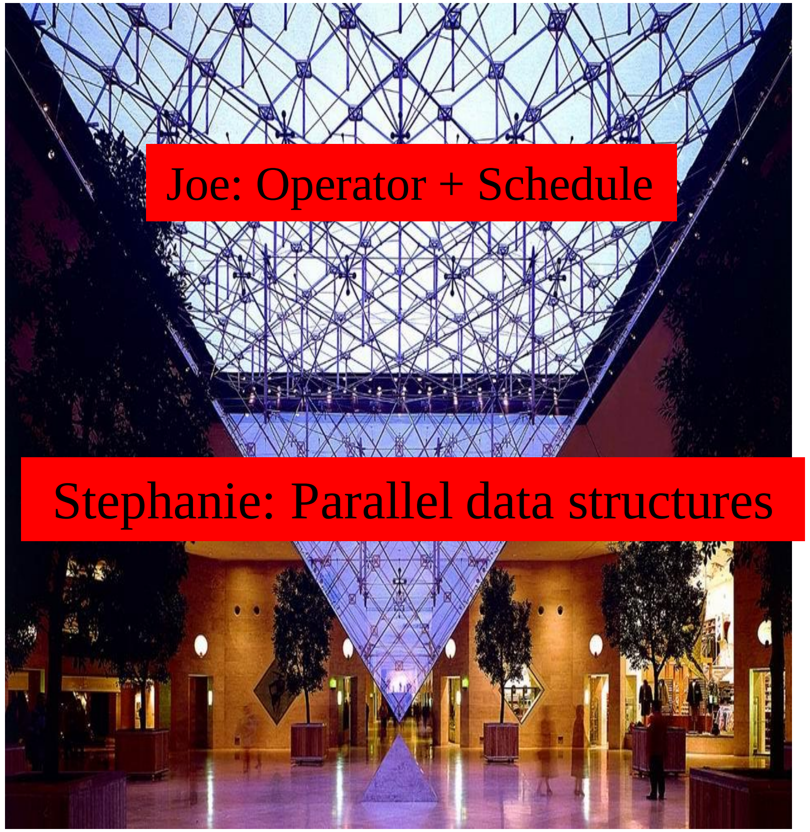


Averages				
VPR 5.0	134.6 seconds		VPR 5.0	134.6 seconds
Galois (1 Thread)	162.4 seconds		Galois (1 Thread)	145.3 seconds
Galois (2 Threads)	106.6 seconds		Galois (2 Threads)	88.8 seconds
Galois (4 Threads)	59.2 seconds		Galois (4 Threads)	43.0 seconds
Galois (8 Threads)	33.7 seconds		Galois (8 Threads)	22.6 seconds

Moctar & Brisk, "Parallel FPGA Routing based on the Operator Formulation" DAC 2014

Conclusions

- Yesterday:
  - Computation-centric view of parallelism
- Today:
  - Data-centric view of parallelism
  - Operator formulation of algorithms
  - Permits a unified view of parallelism and locality in algorithms
  - Joe/Stephanie programming model
  - Galois system is an implementation
- Tomorrow:
  - DSLs for different applications
  - Layer on top of Galois

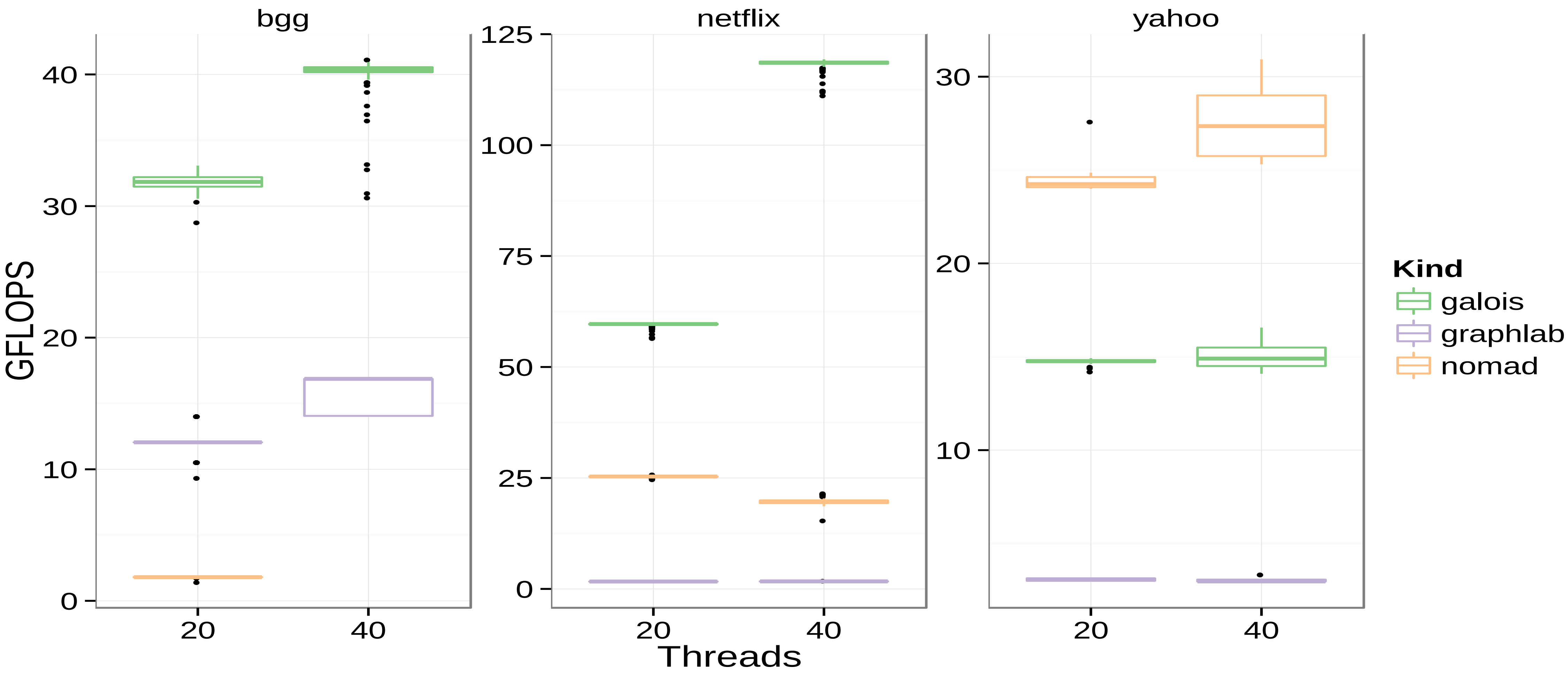


Parallel program = Operator + Schedule + Parallel data structure

More information

- Website
  - http://iss.ices.utexas.edu
- Download
  - Galois system for multicores
  - Lonestar benchmarks
  - All our papers

SGD – Recommender System



nomad with 40 threads on bgg does not converge

Relation to other parallel programming models

- Galois:
  - Parallel program = Operator + Schedule + Parallel data structure
  - Operator can be expressed as a graph rewrite rule on data structure
- Functional languages:
  - Semantics specified in terms of rewrite rules like β-reduction
  - Rules rewrite program, not data structures
- Logic programming:
  - (Kowalski) Algorithm = Logic + Control
  - Control ~ Schedule
- Transactions:
  - Activity in Galois has transactional semantics (atomicity, consistency, isolation)
  - But transactions are synchronization constructs for explicitly parallel languages whereas Joe programming model in Galois is sequential

Save the planet and return your name badge before you leave (on Tuesday)

Microsoft