Charm++ for Productivity and Performance

A Submission to the 2011 HPC Class II Challenge

Laxmikant V. Kale*
Anshu Arya   Abhinav Bhatele   Abhishek Gupta   Nikhil Jain
Pritish Jetley   Jonathan Lifflander   Phil Miller   Yanhua Sun
Ramprasad Venkataraman*   Lukasz Wesolowski   Gengbin Zheng

Parallel Programming Laboratory
Department of Computer Science
University of Illinois at Urbana-Champaign

*{kale, ramv}@illinois.edu
LLNL-PRES-513271

SC11: November 15, 2011

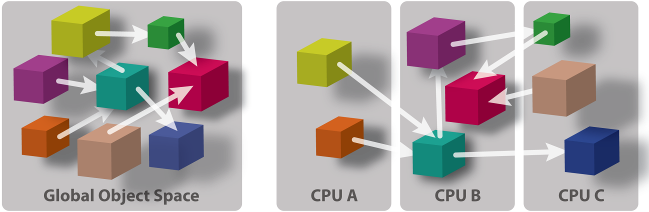
Benchmarks

Required

- Dense LU Factorization
- 1D FFT
- Random Access

Optional

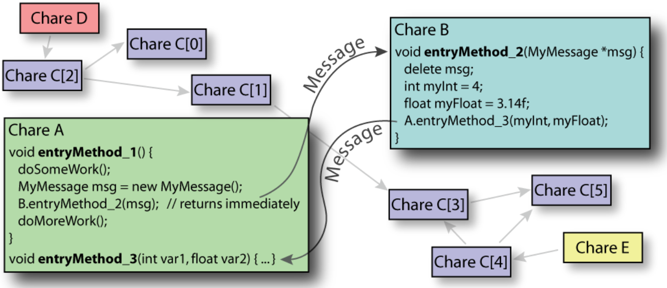
- Molecular Dynamics
- Barnes-Hut



Object-based	Express logic via indexed collections of interacting objects (both data *and* tasks)
Over-decomposed	Expose more parallelism than available processors

Charm++

Programming Model



Runtime-Assisted	scheduling, observation-based adaptivity, load balancing, composition, etc.
Message-Driven	Trigger computation by invoking remote *entry* methods
Non-blocking, Asynchronous	Implicitly overlapped data transfer

- Regular C++ code
  - No special compilers

- Regular C++ code
  - No special compilers
- Small parallel interface description file
  - Can contain control flow DAG
  - Parsed to generate more C++ code

- Regular C++ code
  - No special compilers
- Small parallel interface description file
  - Can contain control flow DAG
  - Parsed to generate more C++ code
- Inherit from framework classes to
  - Communicate with remote objects
  - Serialize objects for transmission
- Exploit modern C++ program design techniques (OO, generics etc)

- Promotes natural expression of parallelism
- Supports modularity

Charm++
Capabilities

- Promotes natural expression of parallelism
- Supports modularity
- Overlaps communication and computation
- Automatically balances load

- Promotes natural expression of parallelism
- Supports modularity
- Overlaps communication and computation
- Automatically balances load
- Automatically handles heterogenous systems
- Adapts to reduce energy consumption
- Tolerates component failures

For more info

http://charm.cs.illinois.edu/why/

Metrics: Performance

Our Implementations in Charm++

Code	Machine	Max Cores	Best Performance
LU	Cray XT5	8K	67.4% of peak
FFT	IBM BG/P	64K	2.512 TFlop/s
RandomAccess	IBM BG/P	64K	22.19 GUPS
MD	Cray XE6	16K	1.9 ms/step (125K atoms)
	IBM BG/P	64K	11.6 ms/step (1M atoms)
Barnes-Hut	IBM BG/P	16K	27×10^9 interactions/s

Metrics: Code Size

Our Implementations in Charm++

Code	C++	CI	Total[1]	Libraries
LU	1231	418	1649	BLAS
FFT	112	47	159	FFTW, Mesh
RandomAccess	155	23	178	Mesh
MD	645	128	773	
Barnes-Hut	2871	56	2927	TIPSY

C++ Regular C++ code

CI Parallel interface descriptions and control flow DAG

[1] Required logic, excluding test harness, input generation, verification, etc.

Metrics: Code Size

Our Implementations in Charm++

Code	C++	CI	Total[1]	Libraries
LU	1231	418	1649	BLAS
FFT	112	47	159	FFTW, Mesh
RandomAccess	155	23	178	Mesh
MD	645	128	773	
Barnes-Hut	2871	56	2927	TIPSY

C++ Regular C++ code

CI Parallel interface descriptions and control flow DAG

Remember: Lots of freebies!

automatic load balancing, fault tolerance, overlap, composition, portability

[1] Required logic, excluding test harness, input generation, verification, etc.

LU: Capabilities

- Composable library
  - Modular program structure
  - Seamless execution structure (interleaved modules)

LU: Capabilities

- Composable library
  - Modular program structure
  - Seamless execution structure (interleaved modules)
- Block-centric
  - Algorithm from a block's perspective
  - Agnostic of processor-level considerations

LU: Capabilities

- Composable library
  - Modular program structure
  - Seamless execution structure (interleaved modules)
- Block-centric
  - Algorithm from a block's perspective
  - Agnostic of processor-level considerations
- Separation of concerns
  - Domain specialist codes algorithm
  - Systems specialist codes tuning, resource mgmt etc

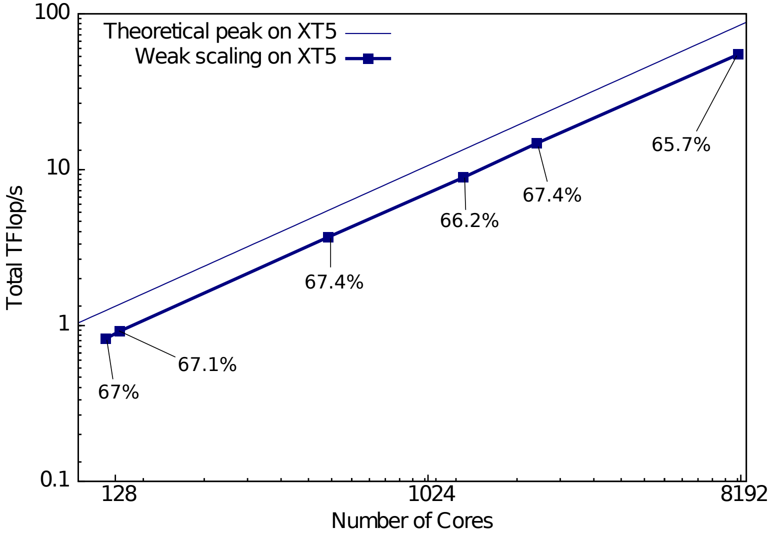
	Lines of Code			Module-specific Commits	
	CI	C++	Total		
Factorization	517	419	**936**	472/572	83%
Mem. Aware Sched.	9	492	501	86/125	69%
Mapping	10	72	82	29/42	69%

LU: Capabilities

- Flexible data placement
  - Experiment with data layout
- Memory-constrained adaptive lookahead

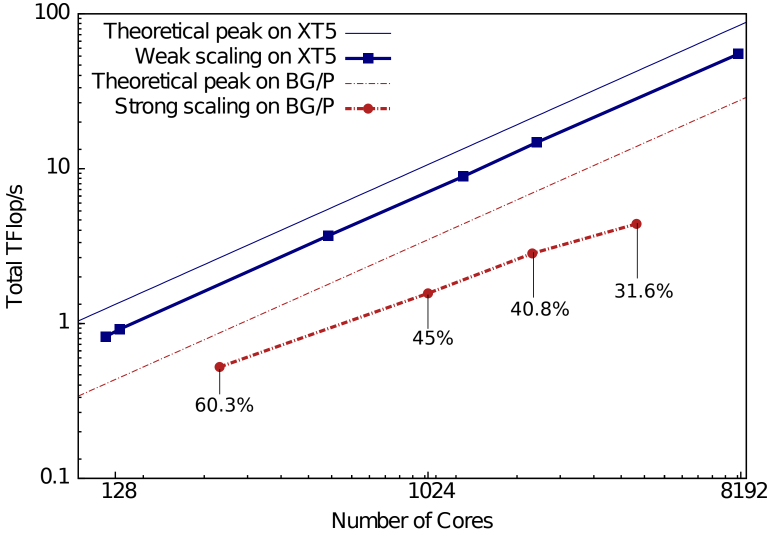
LU: Performance

Weak Scaling: (N such that matrix fills 75% memory)



LU: Performance

... and strong scaling too! (N=96,000)



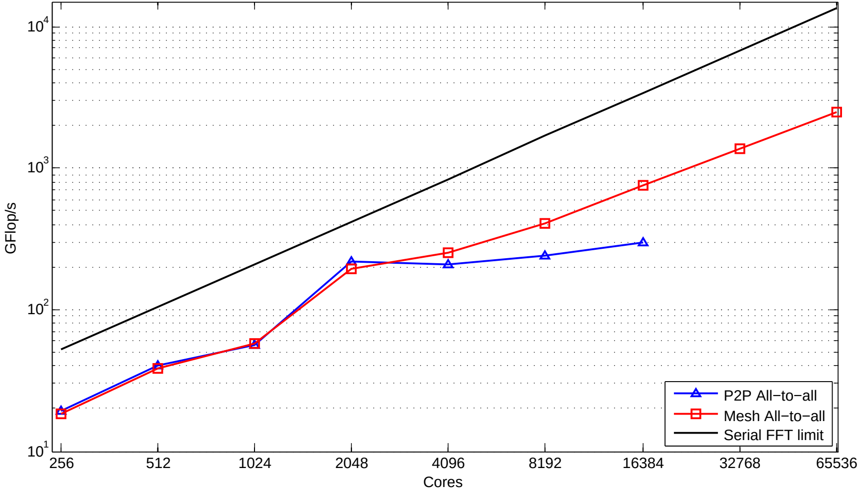
FFT: Parallel Coordination Code

doFFT()

```
for(phase = 0; phase < 3; ++phase) {
    atomic {
        sendTranspose();
    }
    for(count = 0; count < P; ++count)
        when recvTranspose[phase] (fftMsg *msg) atomic {
            applyTranspose(msg);
        }
    if (phase < 2) atomic {
        fftw_execute(plan);
        if(phase == 0)
            twiddle();
    }
}
```

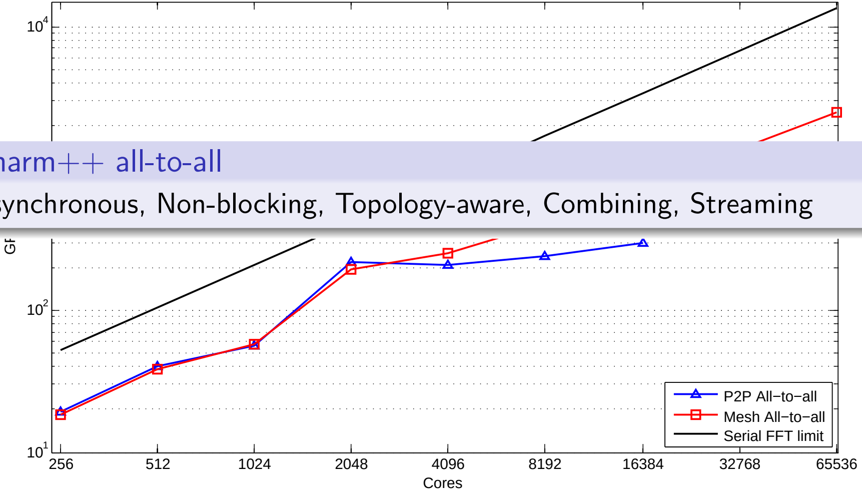
FFT: Performance

IBM Blue Gene/P (Intrepid), 25% memory, ESSL /w fftw wrappers



FFT: Performance

IBM Blue Gene/P (Intrepid), 25% memory, ESSL /w fftw wrappers



Charm++ all-to-all

Asynchronous, Non-blocking, Topology-aware, Combining, Streaming

Random Access

What Charm++ brings to the table

Productivity

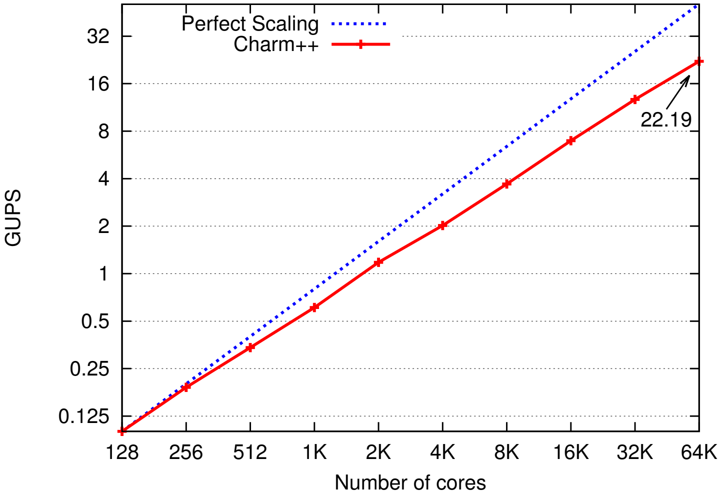
- Automatically detect completion by sensing quiescence
- Automatically detect network topology of partition

Performance

- Uses same Charm++ all-to-all

Random Access: Performance

IBM Blue Gene/P (Intrepid), 2 GB of memory per node

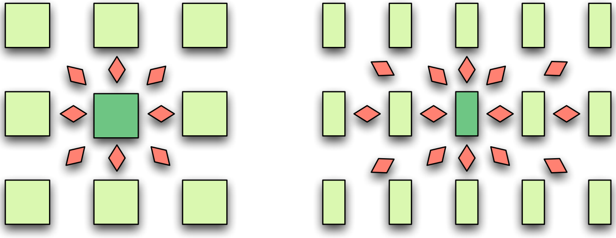


Optional Benchmarks

Why MD and Barnes-Hut?

- Relevant scientific computing kernels
- Challenge the parallelization paradigm
  - Load imbalances
  - Dynamic communication structure
- Express non-trivial parallel control flow

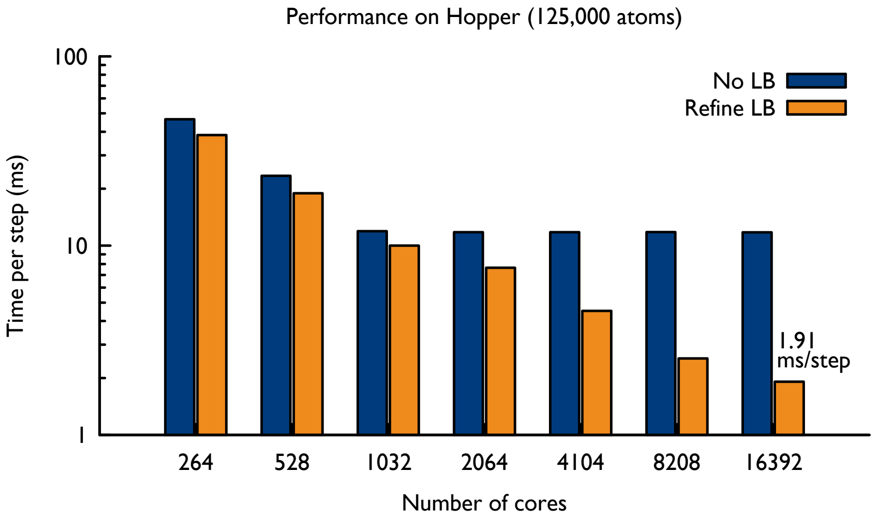
1. Mimics force calculation in NAMD
2. Resembles the miniMD application in the Mantevo benchmark suite
3. SLOC is 773 in comparison to just under 3000 lines for miniMD



(a) 1 Away Decomposition (b) 2 AwayX Decomposition

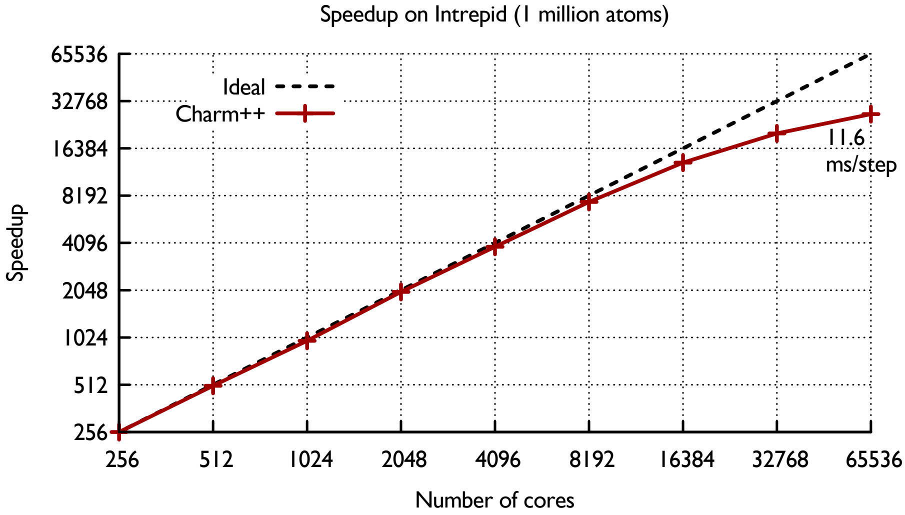
MD: Performance

125,000 atoms. Cray XE6 (Hopper)



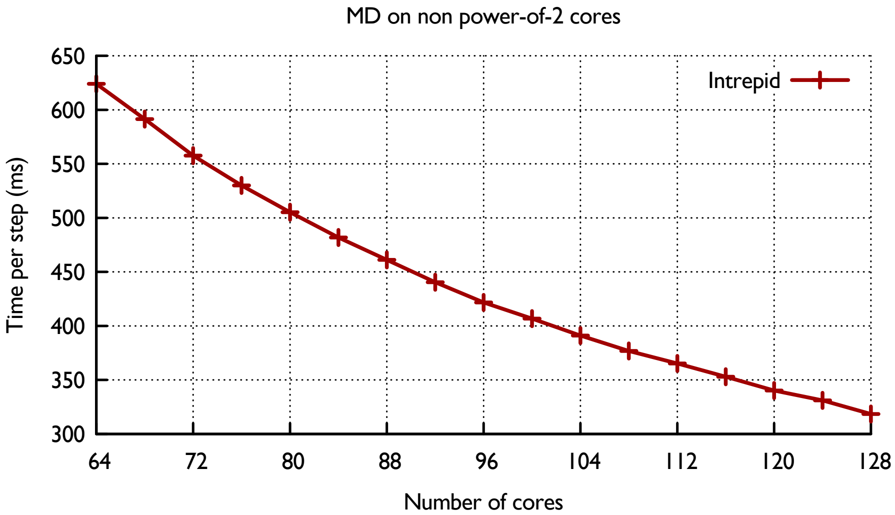
MD: Performance

1 million atoms. IBM Blue Gene/P (Intrepid)



MD: Performance

Number of cores does **not** have to be a power-of-2

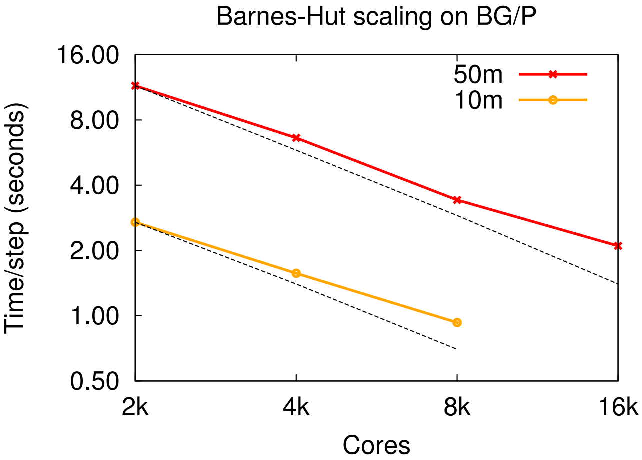


Barnes-Hut: Productivity

1. **Adaptive overlap of computation and communication** allows latency of requests for remote data to be hidden by useful local computation on PEs.
2. **Automatic measurement-based load balancing** allows dissociation of data decomposition from task assignment: balance communication through Oct-decomposition and computation through separate load balancing strategy.

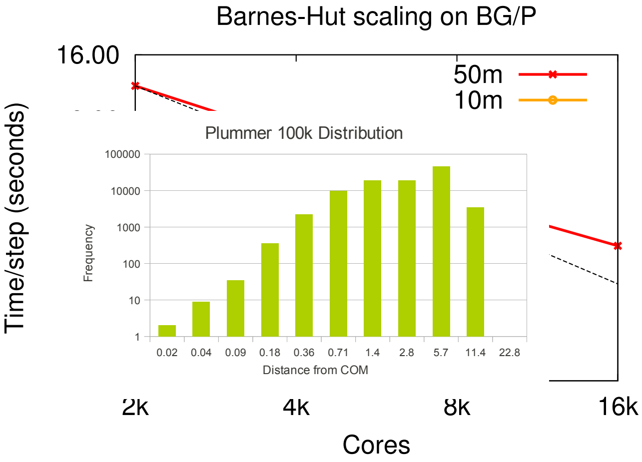
Barnes-Hut: Performance

Non-uniform (Plummer) distribution. IBM Blue Gene/P (Intrepid)



Barnes-Hut: Performance

Non-uniform (Plummer) distribution. IBM Blue Gene/P (Intrepid)



Charm++ at SC11

Temperature-aware load balancing Tue @ 2:00 pm

Fault tolerance protocol PhD Forum; Tue @ 3:45 pm

NAMD at 200K+ cores Thu @ 11:00 am

Topology aware mapping for PERCS Thu @ 4:00 pm

Parallel stochastic optimization Poster

All-to-all simulations on PERCS Poster

For more info

http://charm.cs.illinois.edu/why/

MeshStreamer: Message Routing and Aggregation