# Parallel Zero-Copy Algorithms for Fast Fourier Transform and Conjugate Gradient using MPI Datatypes

**Torsten Hoefler**, Steven Gottlieb

EuroMPI 2010, Stuttgart, Germany, Sep. 13th 2010

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

# Quick MPI Datatype Introduction

- (de)serialize arbitrary data layouts into a message stream
  - Contig., Vector, Indexed, Struct, Subarray, even Darray (HPF-like distributed arrays)
    - Recursive specification possible
  - *Declarative* specification of data-layout
    - "what" and not "how", leaves optimization to implementation (*many unexplored* possibilities!)
  - Arbitrary data permutations (with Indexed)

# Datatype Terminology

- Size
  - Size of DDT signature (total occupied bytes)
  - Important for matching (signatures must match)
- Lower Bound
  - Where does the DDT start
    - Allows to specify “holes” at the beginning
- Extent
  - Size of the DDT
    - Allows to interleave DDT, relatively “dangerous”

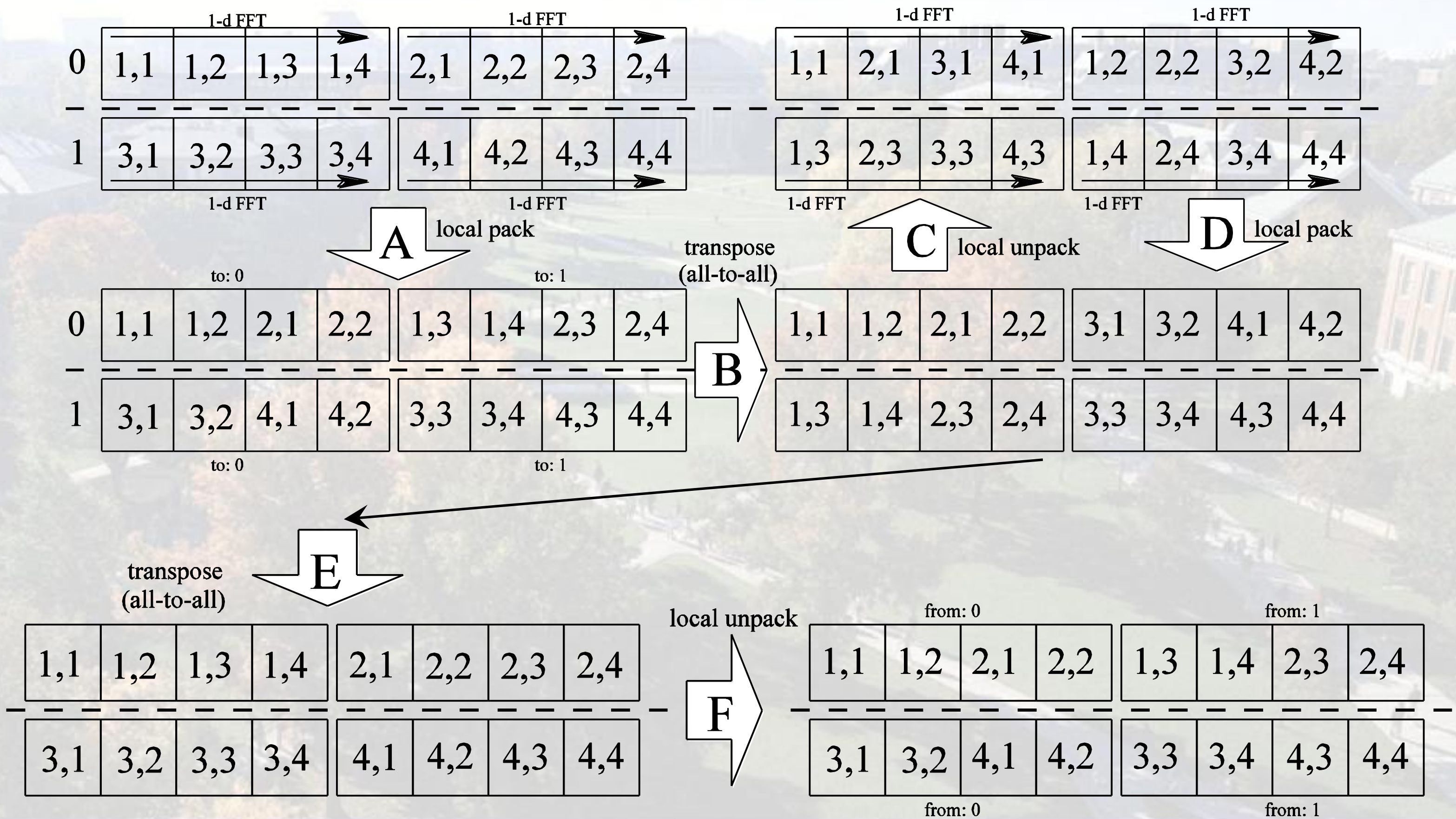
# What is Zero Copy?

- Somewhat weak terminology
  - MPI forces “remote” copy
- But:
  - MPI implementations copy internally
    - E.g., networking stack (TCP), packing DDTs
    - Zero-copy is possible (RDMA, I/O Vectors)
  - MPI applications copy too often
    - E.g., manual pack, unpack or data rearrangement
    - DDT can do both!

# Purpose of this Paper

- Demonstrate utility of DDT in practice
  - Early implementations were bad → folklore
  - Some are still bad → chicken+egg problem
- Show creative use of DDTs
  - Encode local transpose for FFT
- Create realistic benchmark cases
  - Guide optimization of DDT implementations

# 2d-FFT State of the Art



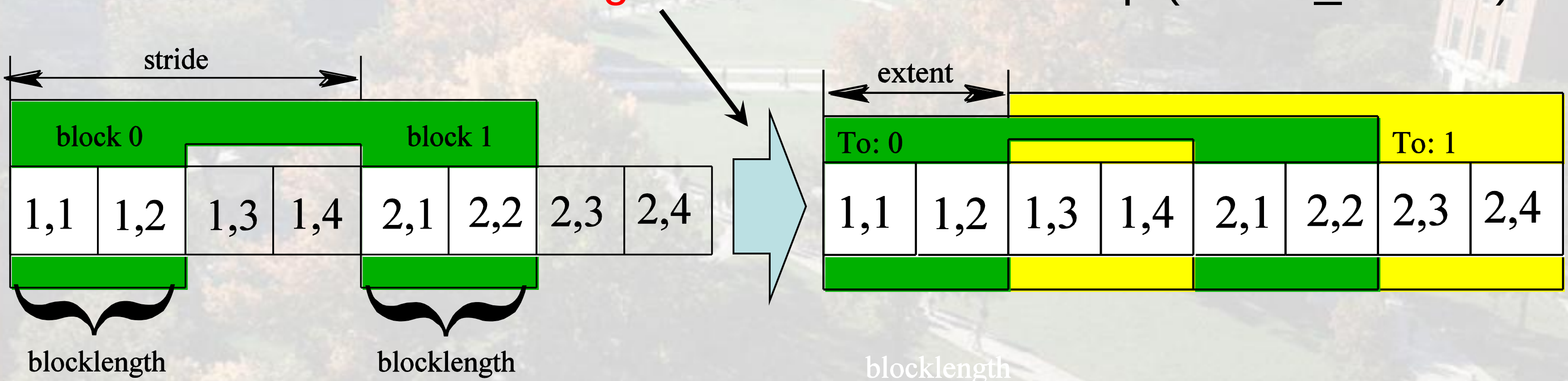
# 2d-FFT Optimization Possibilities

1. Use DDT for pack/unpack (obvious)
   – Eliminate 4 of 8 steps
     • Introduce local transpose

2. Use DDT for local transpose
   – After unpack
   – Non-intuitive way of using DDTs
     • Eliminate local transpose

# The Send Datatype

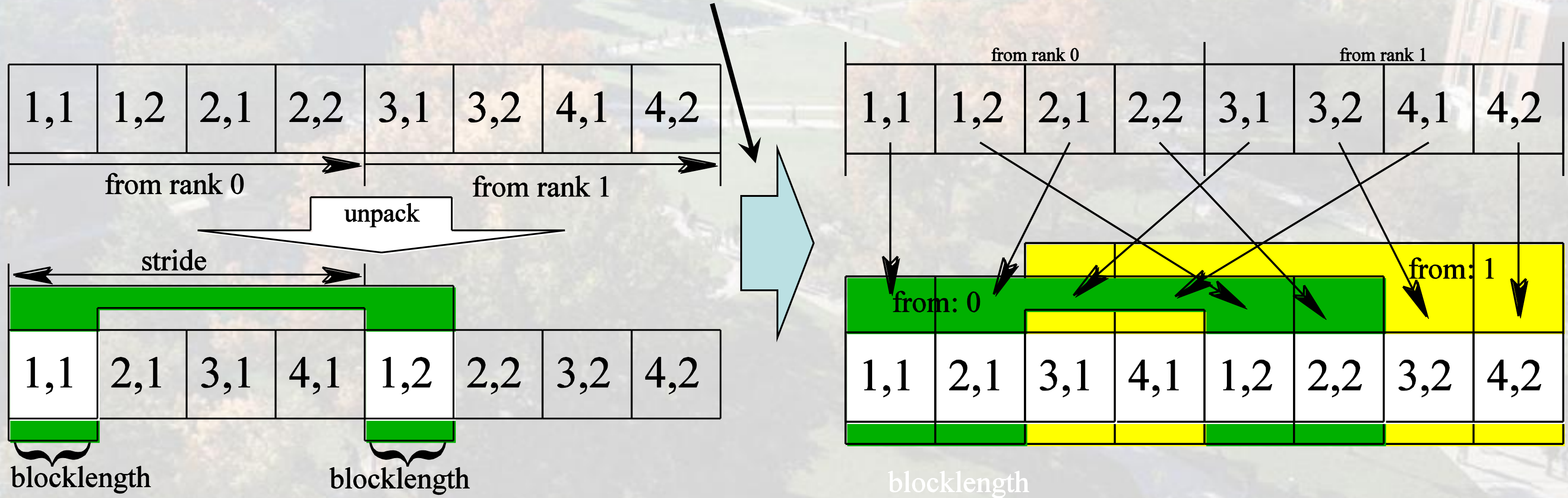
1. Type_struct for complex numbers
2. Type_contiguous for blocks
3. Type_vector for stride
    - Need to change extent to allow overlap (create_resized)




– Three hierarchy-layers

# The Receive Datatype

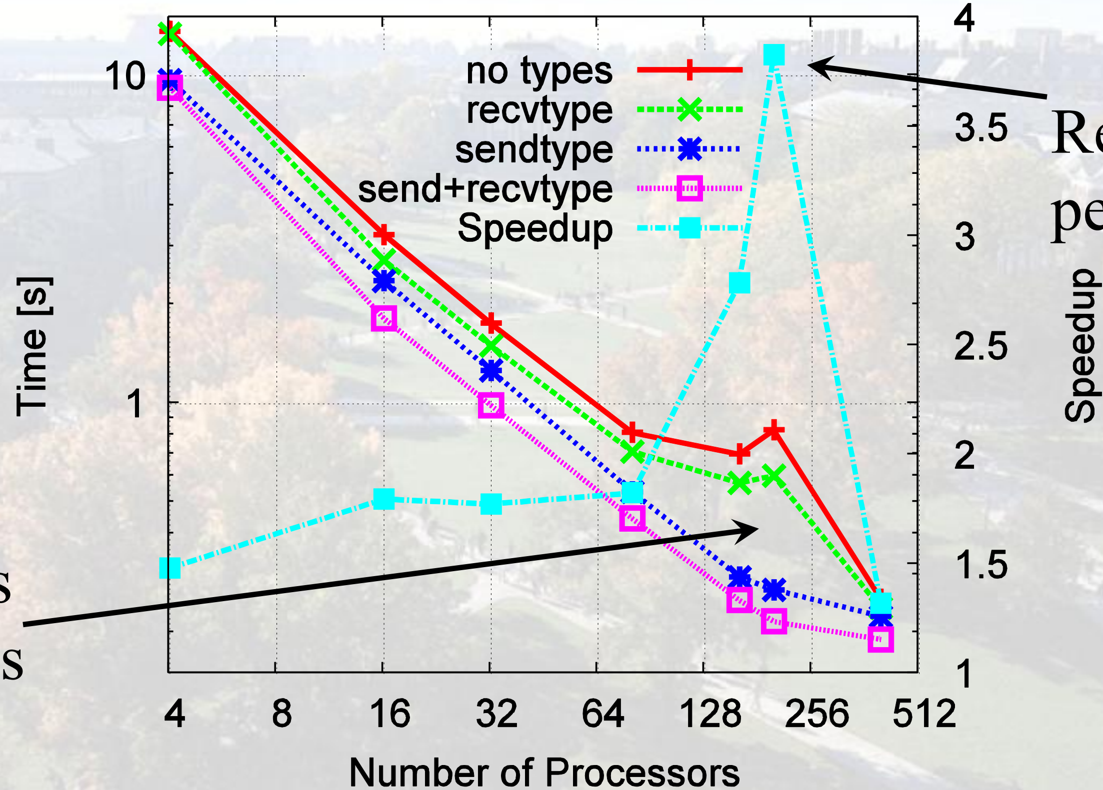
- Type_struct (complex)
- Type_vector (no contiguous, local transpose)
  - Needs to change extent (create_resized)



# Experimental Evaluation

- Odin @ IU
  - 128 compute nodes, 2x2 Opteron 1354 2.1 GHz
  - SDR InfiniBand (OFED 1.3.1).
  - Open MPI 1.4.1 (openib BTL), g++ 4.1.2
- Jaguar @ ORNL
  - 150152 compute nodes, 2.1 GHz Opteron
  - Torus network (SeaStar).
  - CNL 2.1, Cray Message Passing Toolkit 3
- All compiled with “-O3 –mtune=opteron”

# Strong Scaling - Odin ($8000^2$)

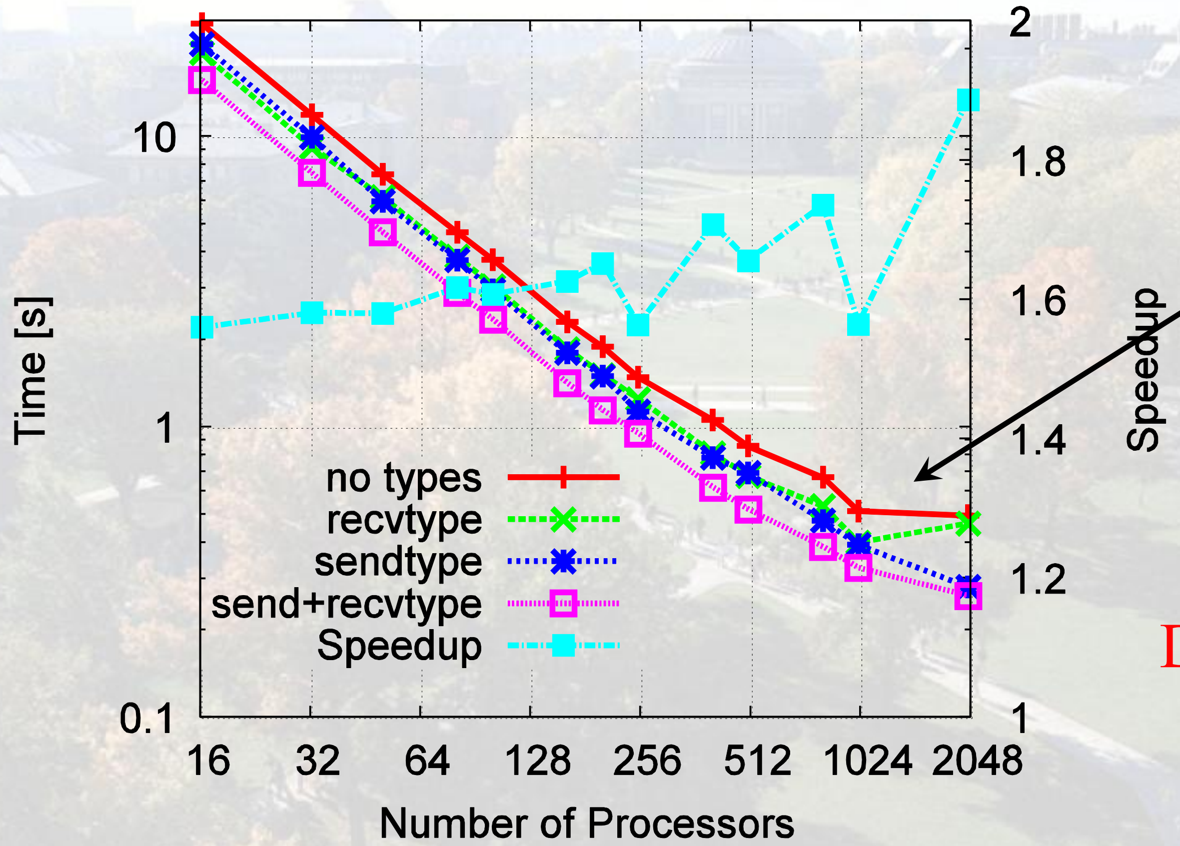



Reproducible peak at P=192

Scaling stops w/o datatypes

- 4 runs, report smallest time, <4% deviation

# Strong Scaling – Jaguar ($20k^2$)




Scaling stops w/o datatypes

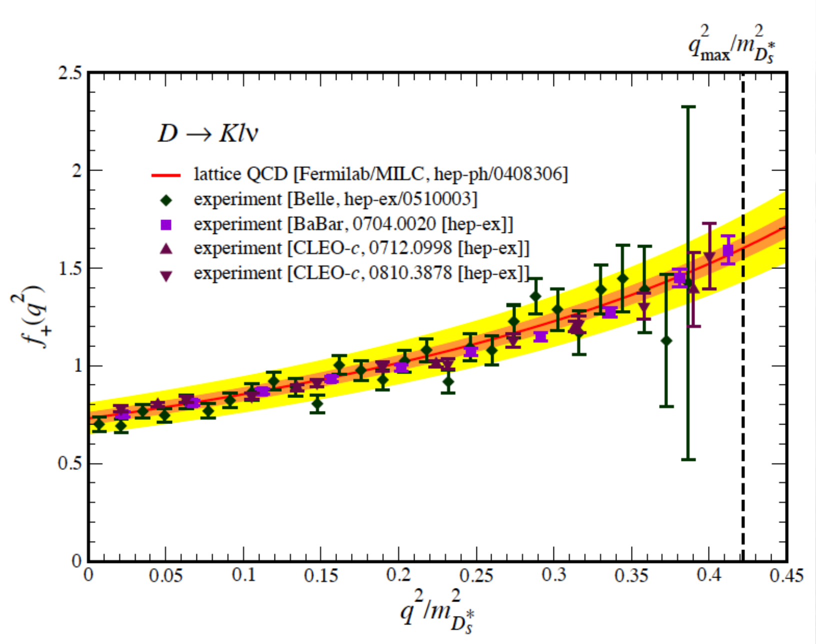
DDT increase scalability

# Negative Results

- Blue Print - Power5+ system
  - POE/IBM MPI Version 5.1
  - Slowdown of 10%
  - Did not pass correctness checks ☹
- Eugene - BG/P at ORNL
  - Up to 40% slowdown
  - Passed correctness check ☺

# Example 2: MIMD Lattice Computation

- Gain deeper insights in fundamental laws of physics
- Determine the predictions of lattice field theories (QCD & Beyond Standard Model)
- Major NSF application
- Challenge:
  - High accuracy (computationally intensive) required for comparison with results from experimental programs in high energy & nuclear physics



# Communication Structure

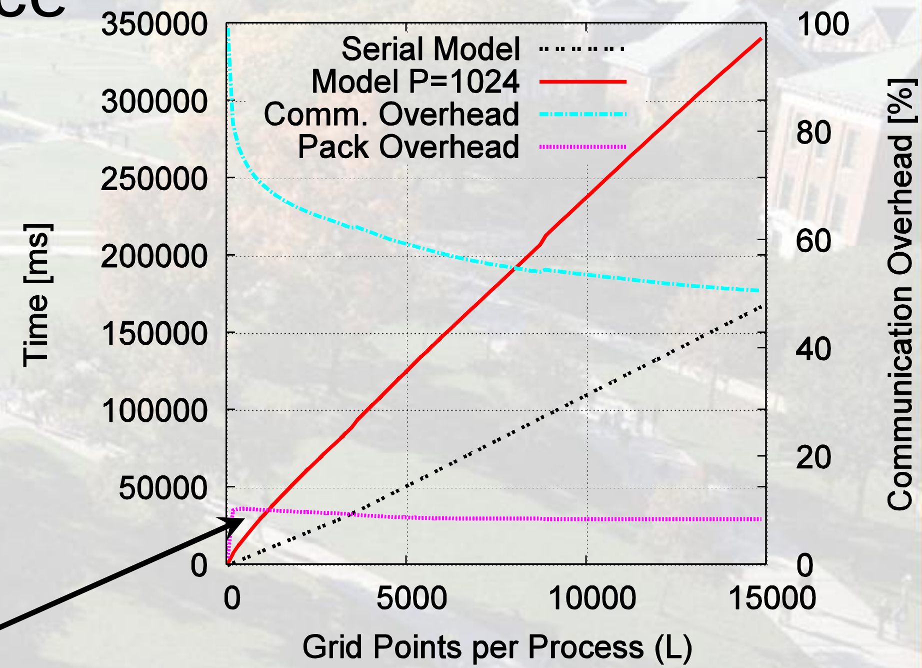
- Nearest neighbor communication
  - 4d array → 8 directions
  - State of the art: manual pack on send side
    - Index list for each element (very expensive)
  - In-situ computation on receive side
- Multiple different access patterns ☹
  - su3_vector, half_wilson_vector, and su3_matrix
  - Even and odd (checkerboard layout)
  - Eight directions
  - 48 contig/hvector DDTs total (stored in 3d array)

# MILC Performance Model

- Designed for Blue Waters
  - Predict performance of 300000+ cores
  - Based in Power7 MR testbed
  - Models manual pack overheads

❖>10% pack time

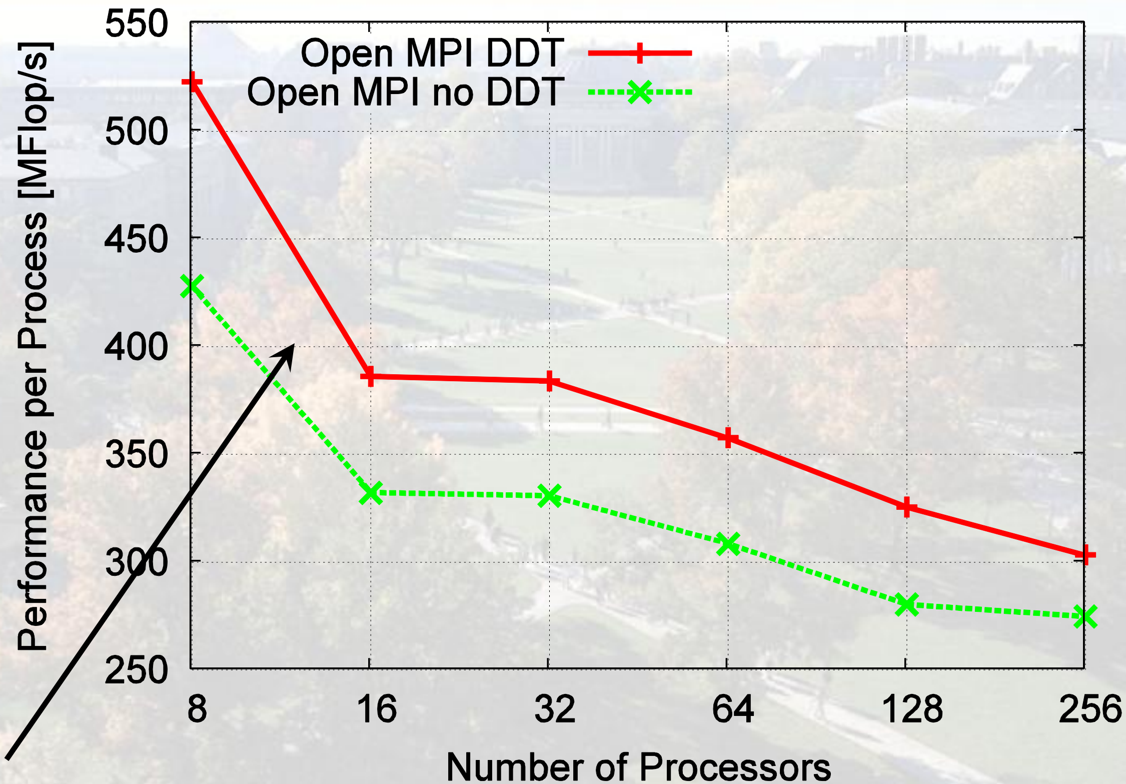
- >15% for small L



# Experimental Evaluation

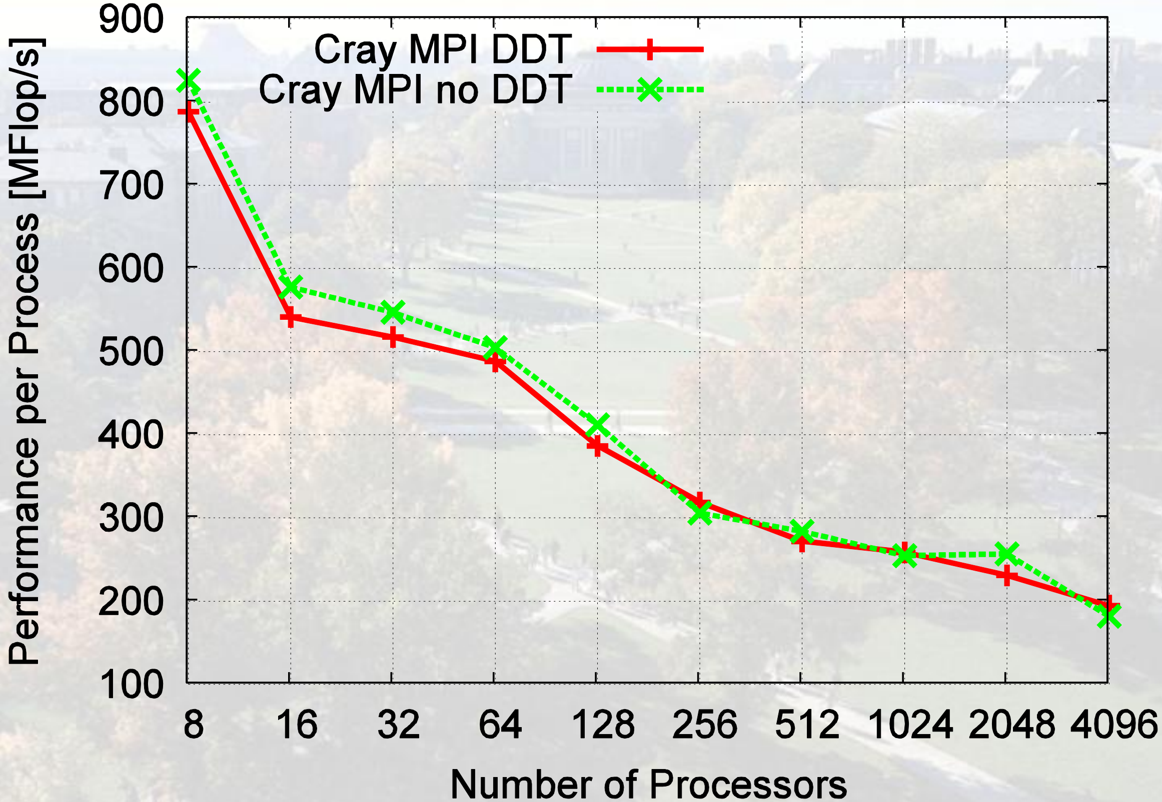
- Weak scaling with $L=4^4$ per process
  - Equivalent to NSF Petascale Benchmark on Blue Waters
- Investigate Conjugate Gradient phase
  - Is the dominant phase in large systems
- Performance measured in MFlop/s
  - Higher is better ☺

# MILC Results - Odin




- 18% speedup!

# MILC Results - Jaguar




- Nearly no speedup (even 3% decrease) ☹

# Conclusions

- MPI Datatypes allow zero-copy
  - Up to a factor of 3.8 or 18% speedup!
  - Requires some implementation effort
- Tool support for datatypes would be great!
  - Declaration and extent tricks make it hard to debug
- Some MPI DDT implementations are slow
  - Some nearly surreal ☺
  - We define benchmarks to solve chicken+egg problem

# Acknowledgments & Support

- Thanks to
  - Bill Gropp
  - Jeongnim Kim
  - Greg Bauer
- Sponsored by

NSF

BLUE WATERS
SUSTAINED PETASCALE COMPUTING

# Backup

## Backup Slides

# 2d-FFT State of the Art

1. perform $N_x/P$ 1-d FFTs in $y$-dimension ($N_y$ elements each)
2. pack the array into a sendbuffer for the all-to-all (A)
3. perform global all-to-all (B)
4. unpack the array to be contiguous in $x$-dimension (each process has now $N_y/P$ $x$-pencils) (C)
5. perform $N_y/P$ 1-d FFTs in $x$-dimension ($N_x$ elements each)
6. pack the array into a sendbuffer for the all-to-all (D)
7. perform global all-to-all (E)
8. unpack the array to its original layout (F)