# UPC Benchmarks

Kathy Yelick
LBNL and UC Berkeley

Joint work with *The Berkeley UPC Group:*

Christian Bell, Dan Bonachea, Wei Chen, Jason Duell, Paul Hargrove, Parry Husbands, Costin Iancu, Rajesh Nishtala, Michael Welcome

# UPC for the High End

**One way to gain acceptance of a new language**

- **Make it run faster than anything else**

**Keys to high performance**

- **Parallelism:**
  - Scaling the number of processors
- **Maximize single node performance**
  - Generate friendly code or use tuned libraries (BLAS, FFTW, etc.)
- **Avoid (unnecessary) communication cost**
  - Latency, bandwidth, overhead
- **Avoid unnecessary delays due to dependencies**
  - Load balance
  - Pipeline algorithmic dependencies

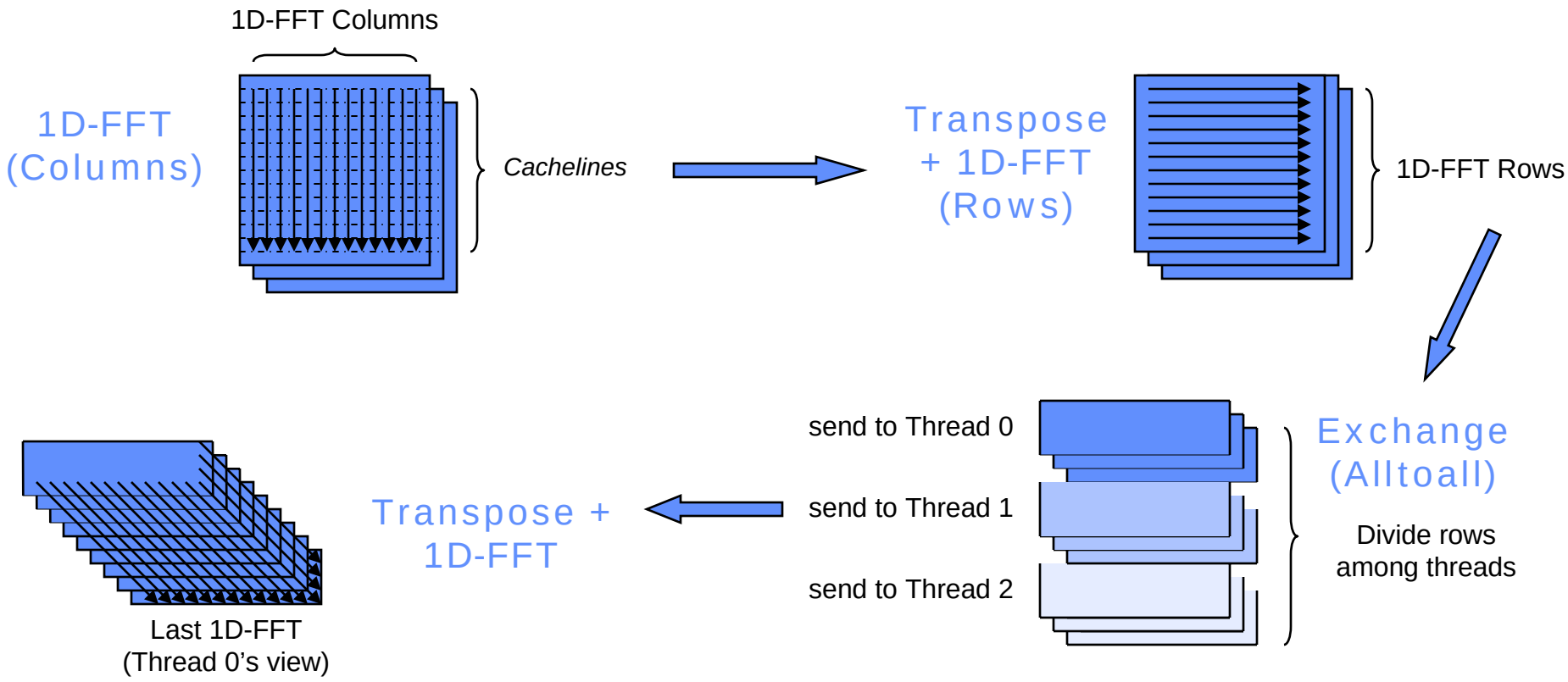
# NAS FT Case Study

- **Performance of Exchange (All-to-all) is critical**
  - Determined by available bisection bandwidth
    - Becoming more expensive as # processors grows
  - Between 30-40% of the applications total runtime
    - Even higher on BG/L scale machine
- **Two ways to reduce Exchange cost**
  1. Use a better network (higher Bisection BW)
  2. Spread communication out over longer period of time:
     "All the wires all the time"

  *Default NAS FT Fortran/MPI relies on #1*

  *Our approach builds on #2*

# 3D FFT Operation with Global Exchange




- **Single Communication Operation (Global Exchange) sends THREADS large messages**
- **Separate computation and communication phases**

# Overlapping Communication

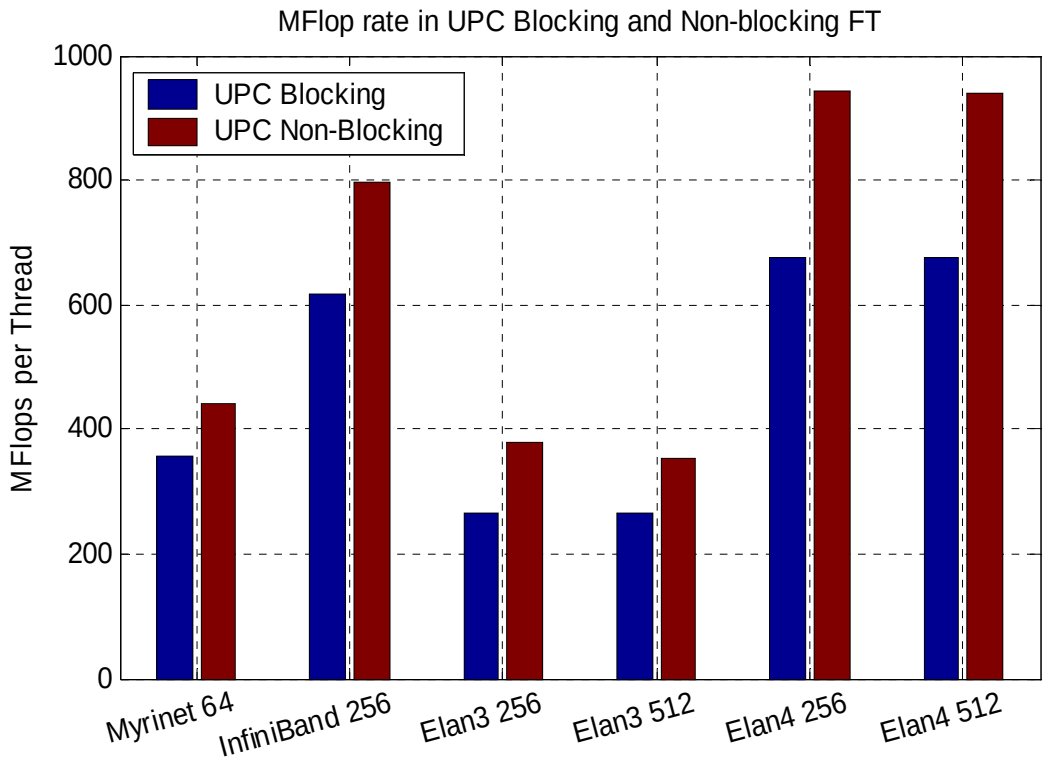
- **Several implementations, each processor owns a set of xy slabs (planes)**
  - 1) Bulk
    - Do column/row FFTs, then send 1/p$^{th}$ of data to each, do z FFTs
    - This can use overlap between messages
  - 2) Slab
    - Do column FFTs, then row FFT on first slab, then send it, repeat
    - When done with xy, wait for and start on z
  - 3) Pencil
    - Do column FFTs, then row FFTs on first row, send it, repeat for each row and each slab

# *Decomposing NAS FT Exchange into Smaller Messages*

- **Example Message Size Breakdown for Class D at 256 Threads**

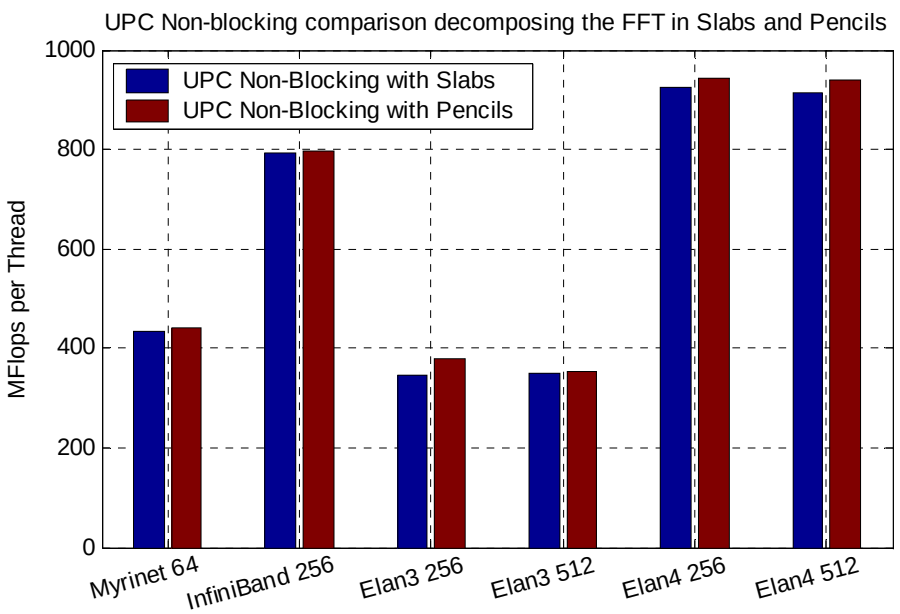
| | |
|---|---|
| **Exchange (Default)** | **512 Kbytes** |
| **Slabs (set of contiguous rows)** | **65 Kbytes** |
| **Pencils (single row)** | **16 Kbytes** |

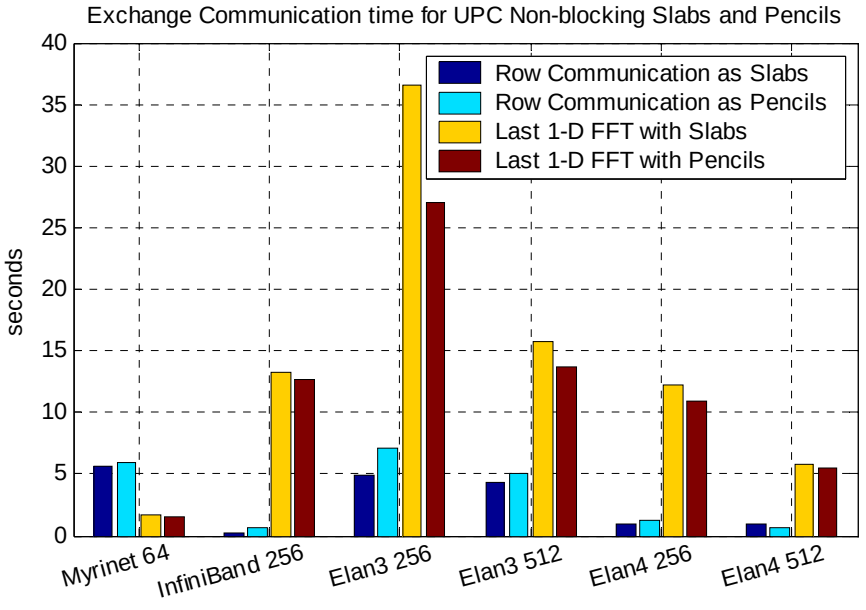
# NAS FT: UPC Non-blocking MFlops




- **Berkeley UPC compiler support non-blocking UPC extensions**
- **Produce 15-45% speedup over best UPC Blocking version**
- **Non-blocking version requires about 30 extra lines of UPC code**

# NAS FT: UPC Slabs or Pencils?




- **In MFlops, pencils (<16Kb messages) are 10% faster**
- **In Communication time, pencils are on average about 15% *slower* than slabs**
- **However, pencils recover more time in allowing for cache-friendly alignment and smaller memory footprint on the last transpose+1D-FFT**

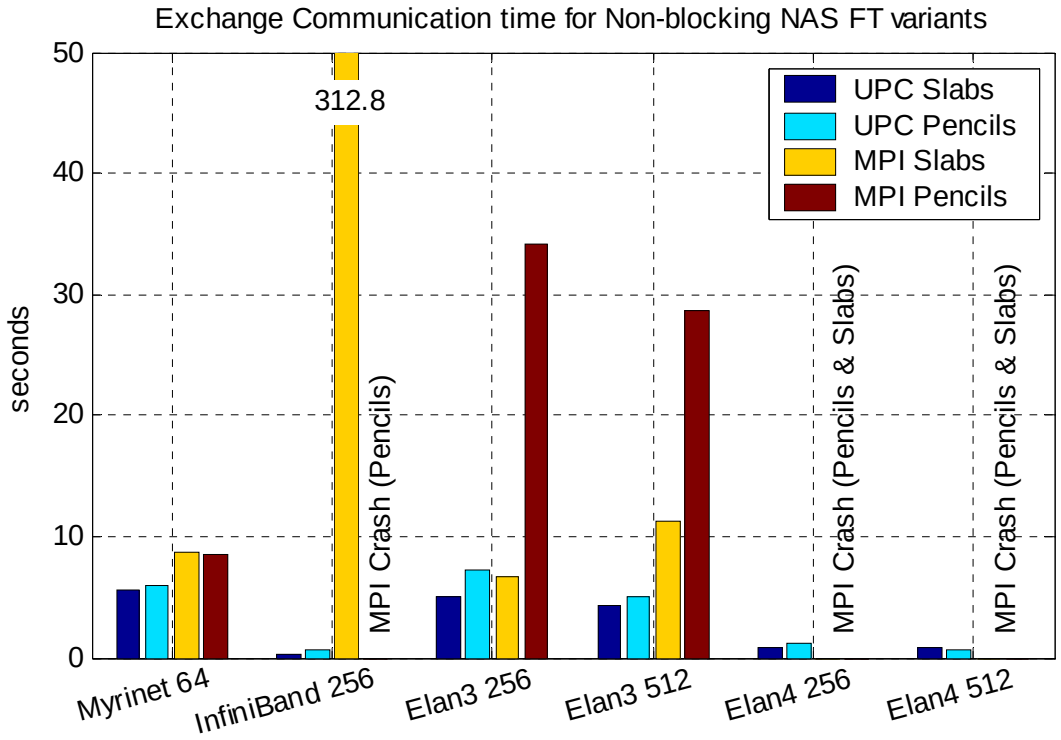
# *Outline*

1. Unified Parallel C (UPC effort at LBNL/UCB)
2. GASNet – UPC's Communications System
   - One-sided communication on Clusters (Firehose)
   - Microbenchmarks
3. Bisection Bandwidth Problem
4. **NAS FT: Decomposing communication to reduce Bisection Bandwidth**
   - Overlapping communication and computation
   - UPC-specific NAS FT implementations
   - **UPC and MPI comparison**

# NAS FT Implementation Variants

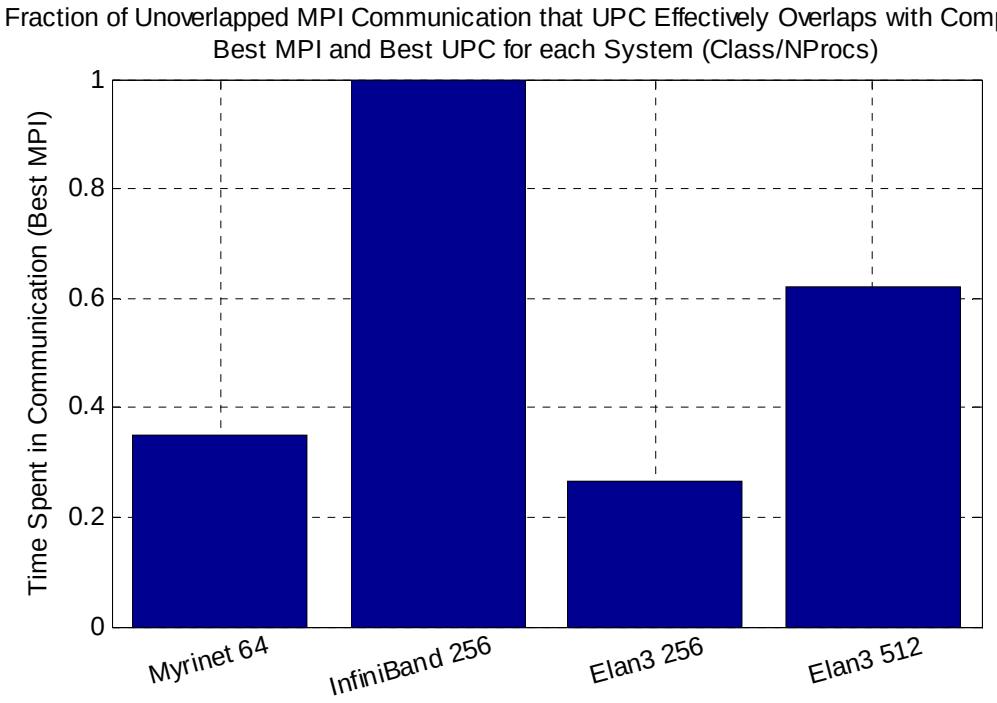
- **GWU UPC NAS FT**
  - Converted from OpenMP, data structures and communication operations unchanged
- **Berkeley UPC NAS FT**
  - Aggressive use of non-blocking messages
    - At Class D/256 Threads, each thread sends 4096 messages with FT-Pencils and 1024 messages with FT-Slabs for each 3D-FFT
  - Use FFTW 3.0.1 for computation (best portability+performance)
  - Add Column pad optimization (up to 4X speedup on Opteron)
- **Berkeley MPI NAS FT**
  - Reimplementation of Berkeley UPC non-blocking NAS-FT
  - Incorporates same FFT and cache padding optimizations
- **NAS FT Fortran**
  - Default NAS 2.4 release (benchmarked version uses FFTW)

# Pencil/Slab optimizations: UPC vs MPI



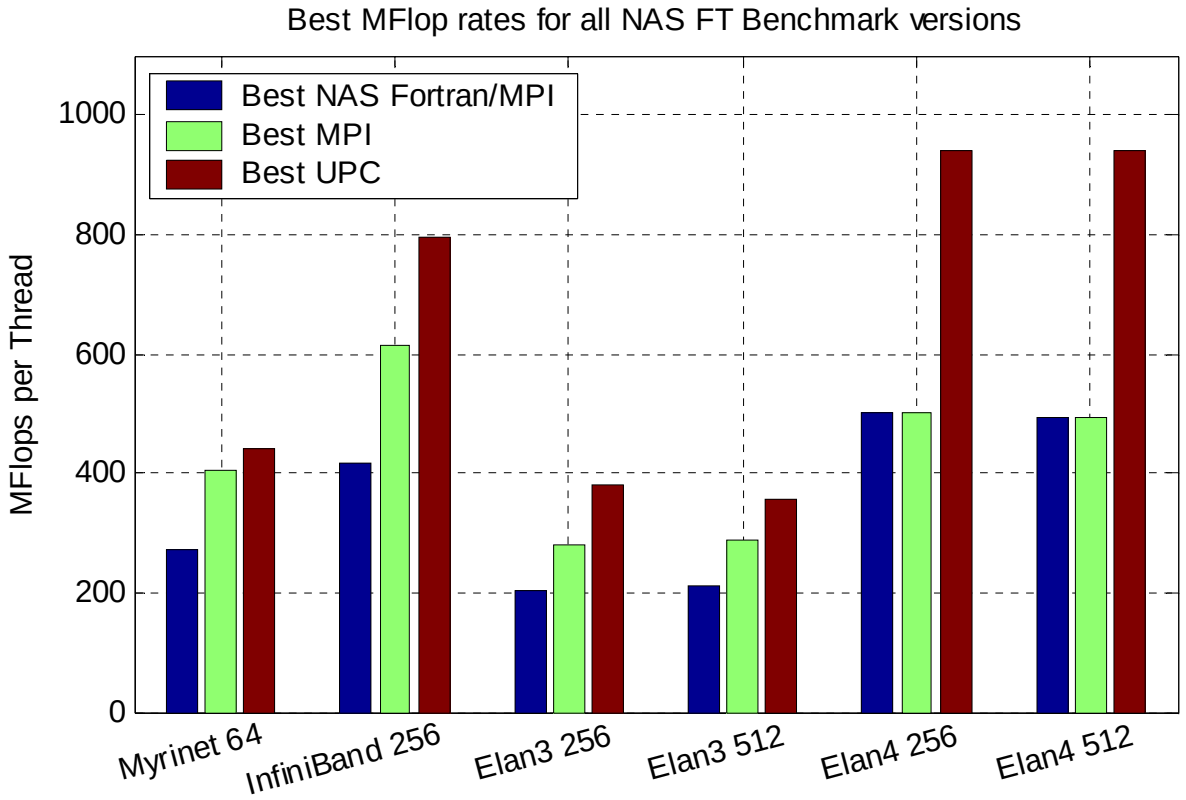

- **Graph measures the cost of interleaving non-blocking communications with 1D-FFT computations**
- **Non-blocking operations are handled uniformly well on UPC but either crash MPI or cause performance problems (with notable exceptions for Myrinet and Elan3)**
- **Pencil communication produces less overhead on the largest Elan4/512 config**

# Pencil/Slab optimizations: UPC vs MPI




- **Same data, viewed in the context of what MPI is able to overlap**
- **"For the amount of time that MPI spends in communication, how much of that time can UPC effectively overlap with computation"**
- **On Infiniband, UPC overlaps almost all the time the MPI spends in communication**
- **On Elan3, UPC obtains more overlap than MPI as the problem scales up**

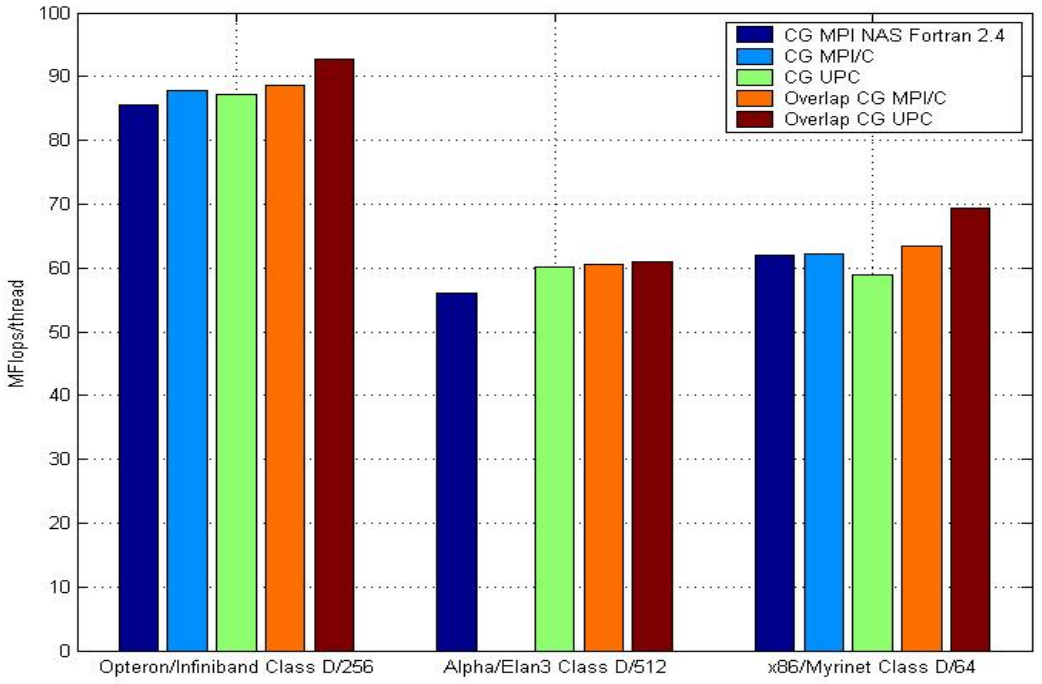
# NAS FT Variants Performance Summary




- **Shown are the largest classes/configurations possible on each test machine**
- **MPI not particularly tuned for many small/medium size messages in flight (long message matching queue depths)**

# Case Study in NAS CG

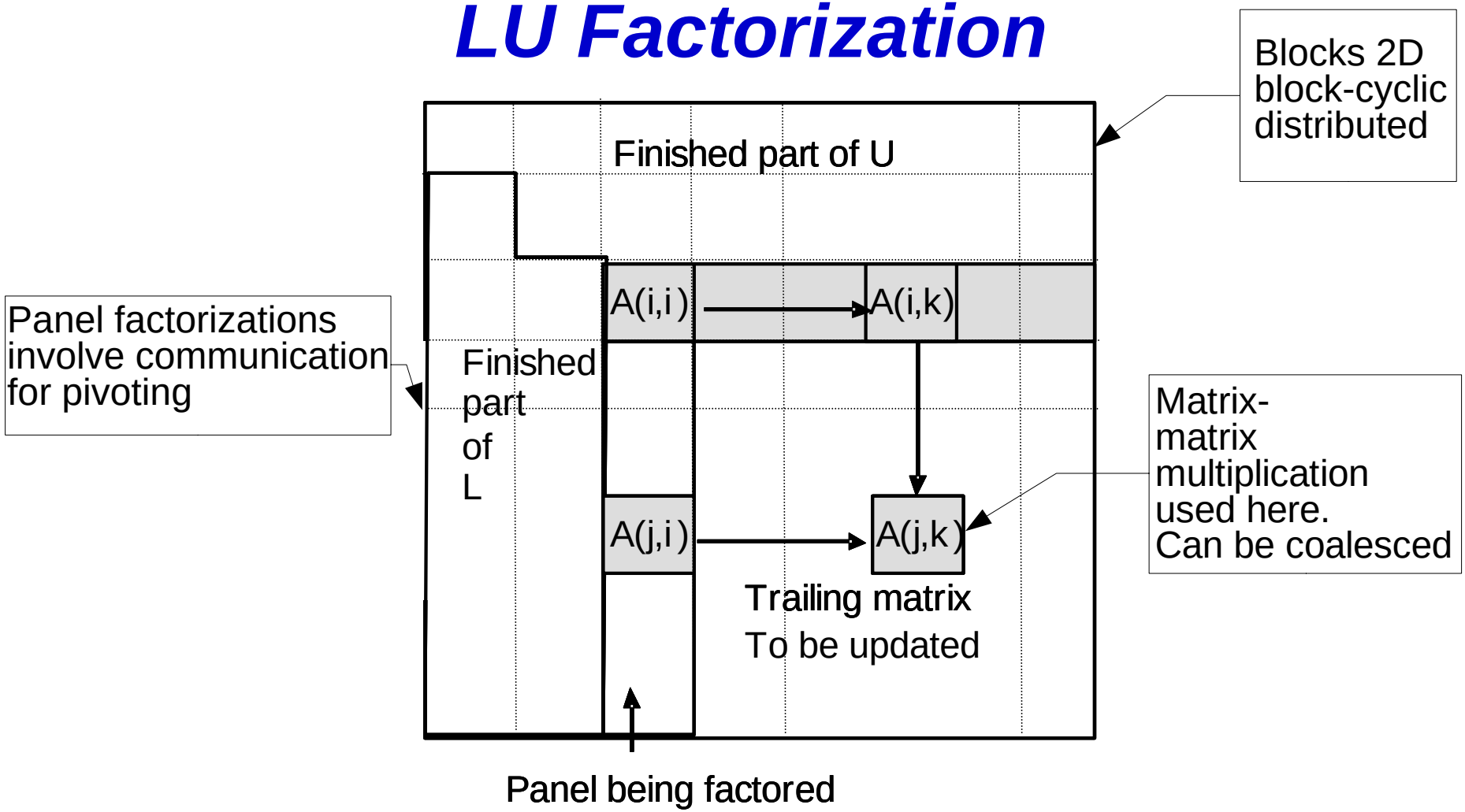
- **Problems in NAS CG are different than FT**
  - Reductions, including vector reductions
  - Highlights need for processor team reductions
- **Using one-sided low latency model**
- **Performance:**
  - Comparable (slightly better) performance in UPC than MPI/Fortran
- **Current focus on more realistic CG**
  - Real matrices
  - 1D layout
  - Optimize across iterations (BeBOP Project)



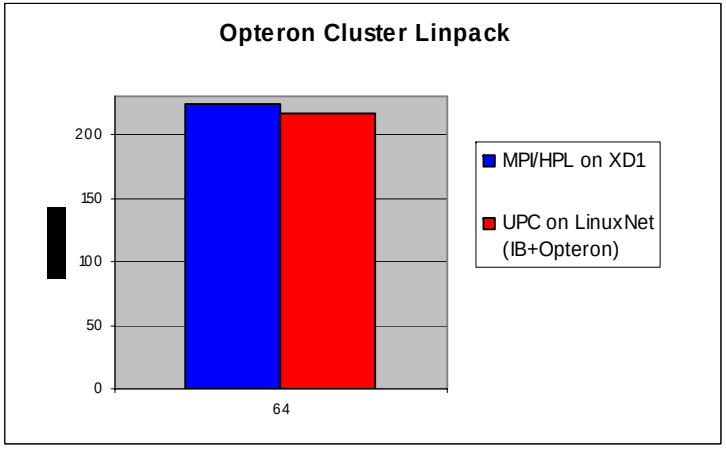
# Direct Method Solvers in UPC

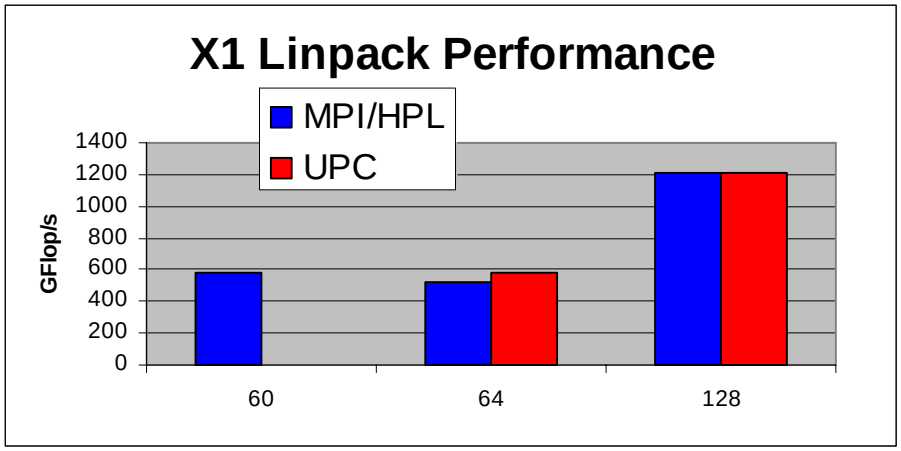
- **Direct methods (LU, Cholesky), have more complicated parallelism than iterative solvers**
  - Especially with pivoting (unpredictable communication)
  - Especially for sparse matrices (dependence graph with holes)
  - Especially with overlap to break dependencies (in HPL, not ScaLAPACK)
- **Today: Complete HPL/UPC**
  - Highly multithreaded: UPC threads + user threads + threaded BLAS
  - More overlap and more dynamic than MPI version for sparsity
  - Overlap limited only by memory size
- **Future: Support for Sparse SuperLU-like code in UPC**
  - Scheduling and high level data structures in HPL code designed for sparse case, but not yet fully “sparsified”

# LU Factorization

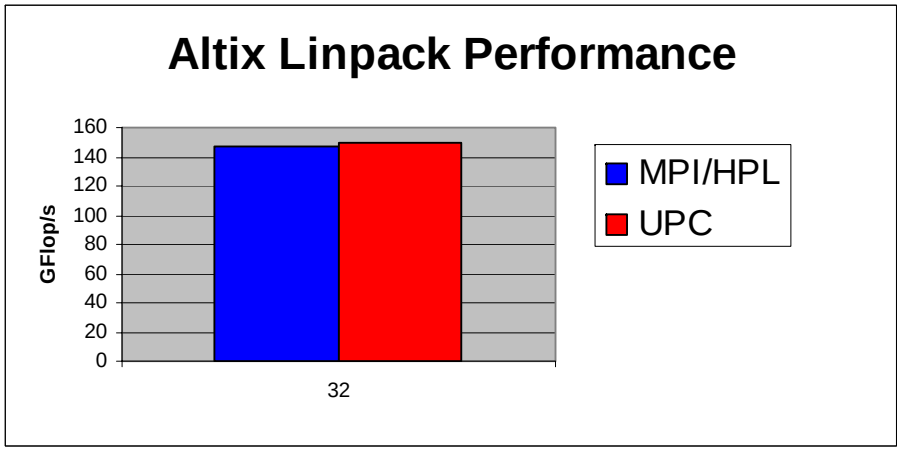


# Dense Matrix HPL UPC vs. MPI




- **Large scaling: 2.2 TFlops on 512 Itanium/Quadrics**




- **Remaining issues**
  - Keeping the machines up and getting large jobs through queues
  - Altix issue with Shmem
  - BLAS on Opteron and X1

# *End of Slides*