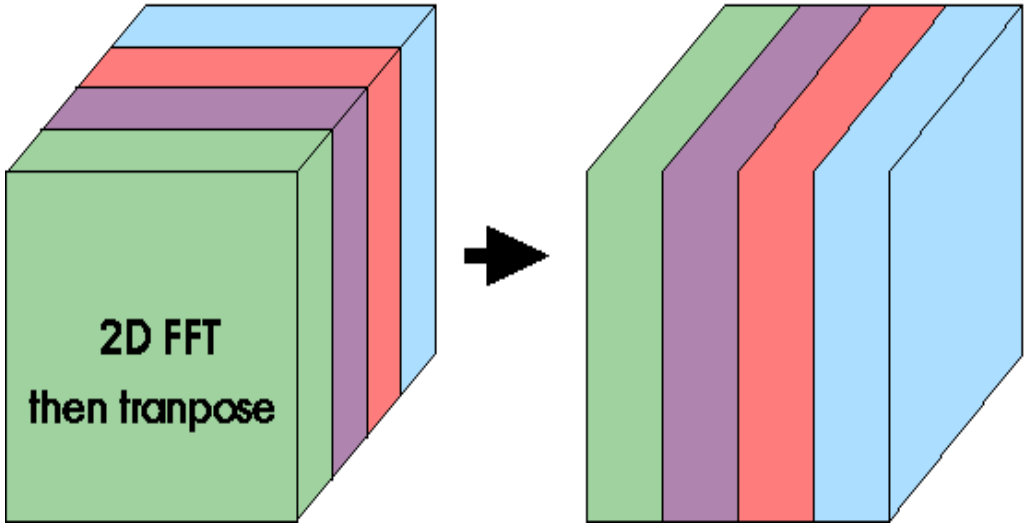
FT Benchmark in UPC

Christian Bell and Rajesh Nishtala

NAS FT Benchmark

- Part of the NAS Parallel Benchmark suite
- Solves a 3-D partial differential equation (PDE) using forward and inverse FFTs
- Important in many spectral-type and engineering codes
- Integrates a heavyweight all-to-all communication step
- Initial exercise was to port the benchmark in UPC and aimed to evaluate UPC for programmability (it is not based on existing GWU NPB benchmarks)

NAS FT Outline



1. Compute 2D FFT of NX*NY over NZ/THREADS planes
2. Send sections of NX*NY/THREADS from each plane to all
3. Transpose each section in Z direction and compute 1D FFT
4. Send results back to original THREADS

NAS FT UPC Implementation

- Each UPC Thread allocates its portion of the 3D Cube in the UPC shared heap
- Little synchronization required – each thread knows where portions of the cube live in the shared heap and issues bulk data puts and gets for 2D and 1D FFTs
- All operations are done in place and in UPC shared memory – no extra memory allocation for temporary storage or double buffering
- 2D FFT plane computations are interleaved with communication (this constitutes a broken up All-to-All).
- Implemented a blocking and non-blocking versions for all communication steps

Existing NAS UPC FT Programmability Comparisons

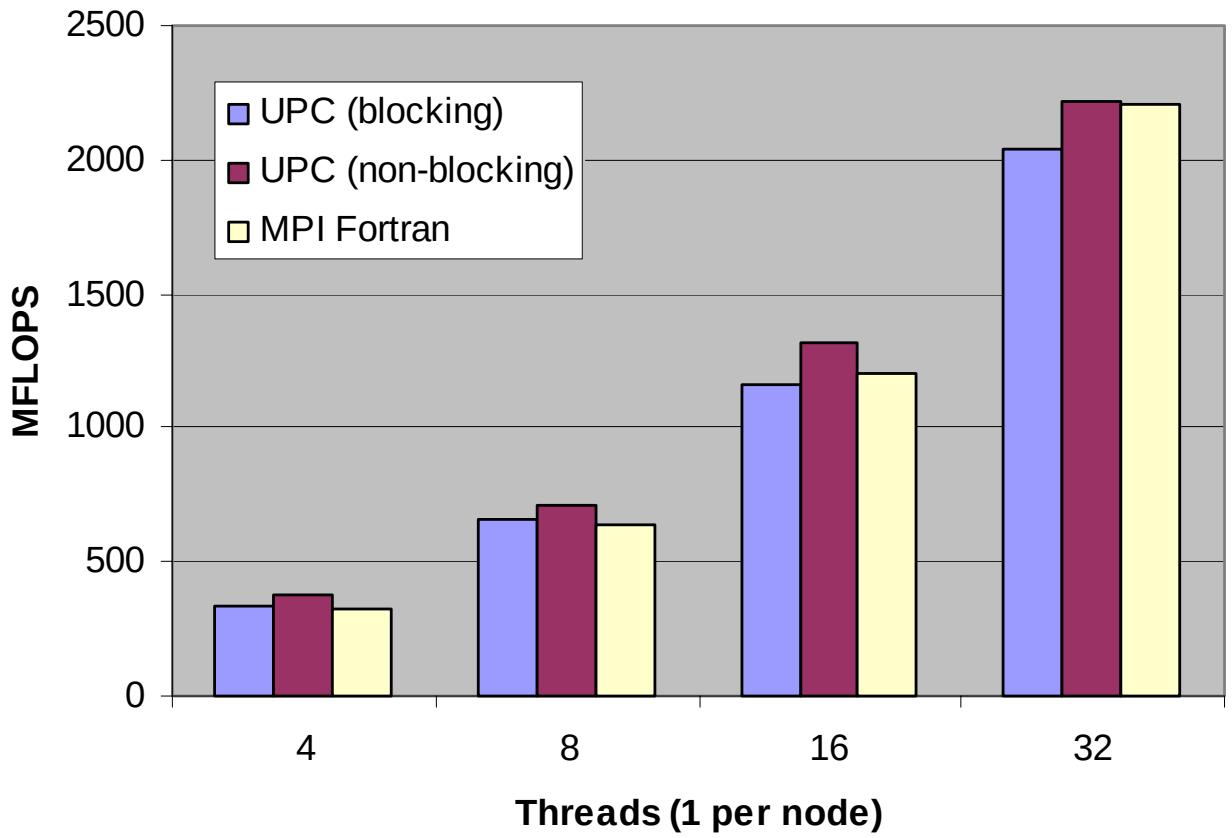
GWU NAS 2.3 FT

- ~600 semi-colons
- Uses 3 x 1D FFTs
- Code history makes the UPC code difficult to read (from Serial Fortran, to OpenMP and finally to UPC)
- Poor programming methodology

Berkeley NAS UPC FT

- ~275 semi-colons
- Uses 2D FFT + 1D FFT
- Strictly programmed in UPC from the ground up
- Improved readability

Performance Results
Berkeley UPC FT vs MPI Fortran FT



80 Dual PIII-866MHz Nodes running Berkeley UPC

(gm-conduit /Myrinet 2K, 33Mhz-64Bit bus)

Conclusion

- Our aim for a highly programmable UPC program also produced a very competitive benchmark
  - A “bulk” type benchmark faster than Fortran MPI
- Communication scheduling between 2D FFT planes circumvents the lack of an efficient All-to-All
- Non-blocking version requires an additional ***17 lines*** of UPC code
  - Makes use of Berkeley UPC specific non-blocking extensions introduced in February 2004
  - MPI Non-blocking semantics require important infrastructure and design changes