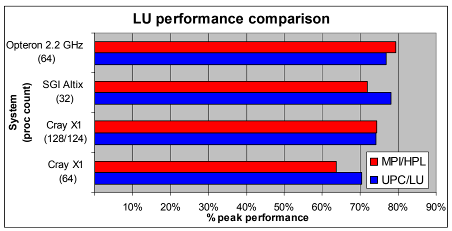
Berkeley UPC Applications

http://upc.lbl.gov

Goals of Application Projects

- Demonstrate that UPC can *outperform* other programming models
  - Take advantage of one-sided communication
  - Show performance advantage *on clusters* with RDMA hardware, as well as shared memory
- Demonstrate *scalability* of UPC
  - NAS FT: *.5 TFlops* on 512p Itanium/Quadrics
  - Linpack: *4.4 Tflops* on 1024p Itanium/Quadrics
- Demonstrate *ease-of-use* on some challenging parallelization problems
  - Delaunay mesh generation
  - Adaptive Mesh Refinement (partially complete)
  - Sparse LU factorization (planned)

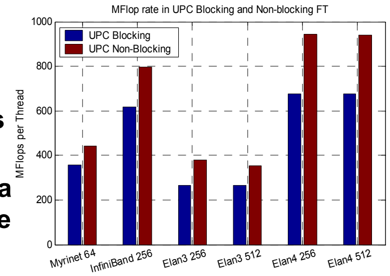
Linpack in UPC



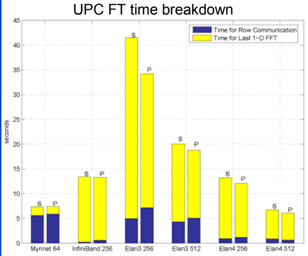
- *UPC Linpack* code is compliant with top500 (HPL)
- Dense case is warm-up for a sparse LU factorization
  - Dependencies, tuning of block sizes, overlap/lookahead are common challenges
  - Differ significantly in compute/communicate ratio
- UPC HPL is less than ½ the code size of MPI HPL
  - Novel multi-threading on SPMD → latency tolerance
  - Memory-constrained lookahead and deadlock avoidance

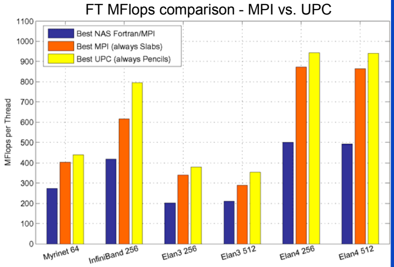
3D FFTs in UPC

- FFT bottleneck is (all-to-all) communication
  - Limited by bisection bandwidth
  - UPC communication has low overhead
  - *Send early and often*: same total data spread over longer period of time to avoid bottleneck
- Bisection bandwidth is increasingly expensive
  → want to use "all the wires all the time"
- Berkeley UPC compiler supports non-blocking bulk memory extensions
  - Non-blocking FT version: ~30 extra lines of UPC code



- Default NAS FT Fortran/MPI sends data all at once: network is idle while processor compute
- UPC implementation overlaps by sending data as it becomes available (per slab or pencil/row)

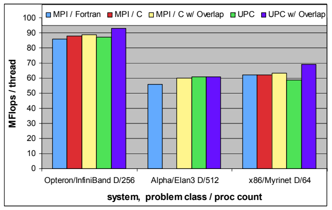




- Slabs win in MPI: overlap is good, but fine-grained overlap less effective due to high msg overheads
- Pencils win in UPC: low overhead + benefit of better local memory locality (smaller messages)

Conjugate Gradient in UPC

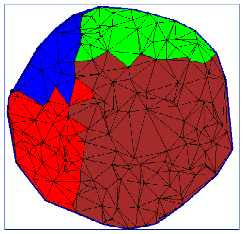
- CG: Iterative sparse solver w/ Sparse Matrix-Vector Multiply (SPMV)
- 2D (NAS-optimized) and 1D partitioned versions



- Bottleneck is reductions, which are *latency-limited*
- UPC version overlaps multi-word reductions with the local SPMV computations
- Outperforms MPI version by up to 10%

Mesh Generation in UPC

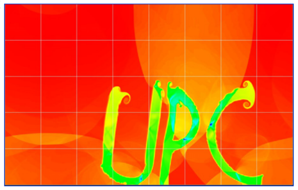
- 2D Delaunay triangulation based on "Triangle" software



- Parallel version:
  - Dynamic load balancing
  - App-level software caching
  - Parallel sorting

Fluid Dynamics

- Finite difference hyperbolic solver in UPC
  – Numerics in FORTRAN*
  – Data/control structures in UPC
- Warm-up for Adaptive Mesh Refinement (AMR)
- Mach 2 wave in a 2-D periodic chamber with a dense fluid in the shape of the letters: U P C



*Thanks to the ANAG group at LBL