The Berkeley UPC Project
http://upc.lbl.gov

Berkeley UPC Compiler Features
- Translates UPC to C w/calls to Berkeley runtime
- Fully compliant with UPC 1.1.1 specification
- Complete implementation of UPC collectives & I/O
- Entirely open source, based on Open64
- Berkeley upc_mem*() library extensions for non-blocking and non-contiguous mem. copies

Berkeley UPC Runtime
- Fully open source, well-documented interface
- Berkeley GASNet used for communication:
  - Portability from layered design
  - Performance from inline functions, macros, and network-specific implementations
  - Collectives implemented using GASNet 2 support:

Berkeley UPC Applications
- Application benchmark suite
- Used to evaluate ease-of-use, performance, and scalability
- Motivates language/compiler extensions

Compiler Optimizations

- upc_forall Affinity Test Removal
  - Removes the runtime branch from upc_forall loops
  - Works for integer and shared address affinity expressions

- Split-Phase Access
  - Transforms relaxed shared refs into split-phase comm.
  - Moves reads initiations up, write completions down
  - Conforms to the UPC memory consistency model
  - Converts fine-grained accesses into bulk msgs
  - Currently supports 1D arrays, extending to multi-D

- Message Coalescing
  - Message Coalescing for Matrix-Vector Multiply

Interoperability
- Berkeley UPC provides app interoperability:
  - UPC calls to/from C, C++, FORTRAN, MPI
- Runtime supports multiple UPC compilers:
  - Berkeley source-to-source UPC translator
  - Intrepid GCC/UPC binary compiler

Tools and Testing
- Debugging with Etnus Totalview upcoming
- Extensive compliance test suite (>500 tests)
- GASNet Trace performance analysis tool
  - Reports time threads spend waiting at barriers
  - Summarizes shared memory refs, local & remote
  - Reveals load imbalance, communication "leaks"
  - message size stats, all by source line & thread:

NAS MG
- Uses one-sided programming style
- Scales well to 1,000's of UPC threads

Godunov CFD
- Interfaces to FORTRAN numerics from LBNL Chombo
- Uses multi-D strided data movement
- Motivates strided memcpy extensions

Parallel Delaunay Triangulation
- Demonstrates use of irregular distributed data structures
  - Directories maintain replicated cached copies of remote data
  - App-level teams for collectives and memory allocation
- Hand optimizations motivate compiler support for message coalescing and strip-mining

UPC Linpack
- Dense LU factorization (HPL)
- First step in sparse direct factorization (SuperLU)

The Berkeley UPC Project
http://upc.lbl.gov

Berkeley UPC Compiler Features
- Translates UPC to C w/calls to Berkeley runtime
- Fully compliant with UPC 1.1.1 specification
- Complete implementation of UPC collectives & I/O
- Entirely open source, based on Open64
- Berkeley upc_mem*() library extensions for non-blocking and non-contiguous mem. copies

Berkeley UPC Runtime
- Fully open source, well-documented interface
- Berkeley GASNet used for communication:
  - Portability from layered design
  - Performance from inline functions, macros, and network-specific implementations
  - Collectives implemented using GASNet 2 support:

Berkeley UPC Applications
- Application benchmark suite
- Used to evaluate ease-of-use, performance, and scalability
- Motivates language/compiler extensions

Compiler Optimizations

- upc_forall Affinity Test Removal
  - Removes the runtime branch from upc_forall loops
  - Works for integer and shared address affinity expressions

- Split-Phase Access
  - Transforms relaxed shared refs into split-phase comm.
  - Moves reads initiations up, write completions down
  - Conforms to the UPC memory consistency model
  - Converts fine-grained accesses into bulk msgs
  - Currently supports 1D arrays, extending to multi-D

- Message Coalescing
  - Message Coalescing for Matrix-Vector Multiply

Interoperability
- Berkeley UPC provides app interoperability:
  - UPC calls to/from C, C++, FORTRAN, MPI
- Runtime supports multiple UPC compilers:
  - Berkeley source-to-source UPC translator
  - Intrepid GCC/UPC binary compiler

Tools and Testing
- Debugging with Etnus Totalview upcoming
- Extensive compliance test suite (>500 tests)
- GASNet Trace performance analysis tool
  - Reports time threads spend waiting at barriers
  - Summarizes shared memory refs, local & remote
  - Reveals load imbalance, communication "leaks"
  - message size stats, all by source line & thread:

NAS MG
- Uses one-sided programming style
- Scales well to 1,000's of UPC threads

Godunov CFD
- Interfaces to FORTRAN numerics from LBNL Chombo
- Uses multi-D strided data movement
- Motivates strided memcpy extensions

Parallel Delaunay Triangulation
- Demonstrates use of irregular distributed data structures
  - Directories maintain replicated cached copies of remote data
  - App-level teams for collectives and memory allocation
- Hand optimizations motivate compiler support for message coalescing and strip-mining

UPC Linpack
- Dense LU factorization (HPL)
- First step in sparse direct factorization (SuperLU)