



Berkeley UPC Runtime Status Report

September 21, 2004

Jason Duell
LBL & UC Berkeley



Topics



- **Pthreaded runtime**
- **Support for GCCUPC (Intrepid)**
- **C++/MPI Interoperability**
- **Usability/stability improvements**
- **Future work**



Pthreaded UPC



- **Pthreaded version of the runtime**
 - Our current strategy for SMPs and clusters of SMPs
- **Implementation challenge: thread-local data.**
 - Different solution for binary vs. source-to-source
- **Has exposed issues in UPC specification:**
 - Global variables in C vs. UPC
 - Misc. standard library issues: rand() behavior



Pthreads vs. SysV Shared Memory



- **Future work: implement System V shared memory, and compare to pthreads**
 - **Benefit: many scientific libraries are not pthread-safe.**
 - **But: bootstrapping issues, limits on size of shared regions**
- **Pthreads share a single network connection:**
 - **Fewer network points for fixed number of UPC threads**
 - **Any pthread can service pending requests for all: better network attentiveness**
 - **But SysV shared memory may avoid lock contention within network API.**



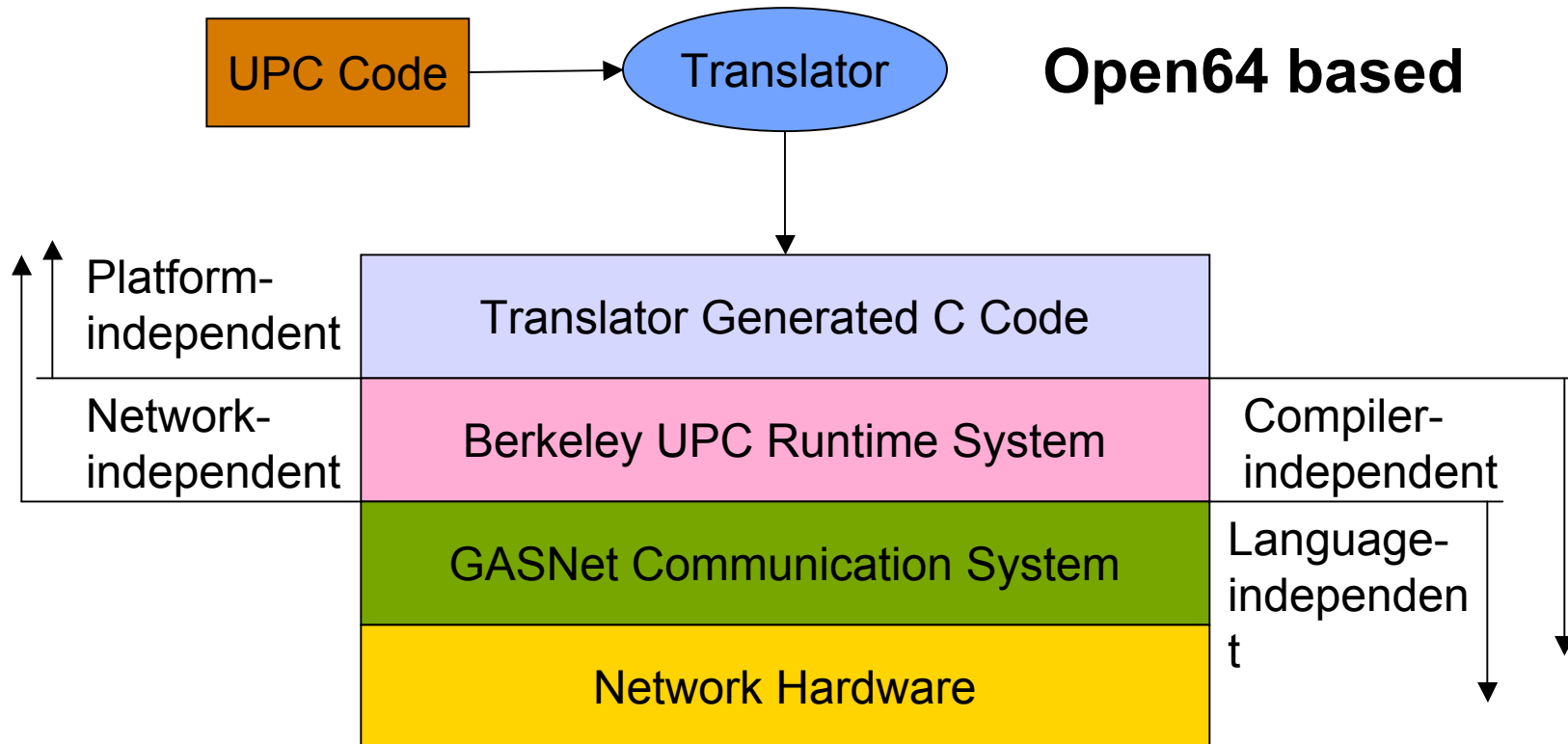
Topics



- **Pthreaded runtime**
- **Support for GCCUPC (Intrepid)**
- **C++/MPI Interoperability**
- **Usability/stability improvements**
- **Future work**



Berkeley UPC Compiler



Two Goals: **Portability** and **High-Performance**



GCCUPC (Intrepid) support



- **GCCUPC can now use Berkeley UPC runtime**
 - Generates binary objects that link with our library.
- **GCCUPC previously only for shared memory: now able to use any GASNet network**
 - Myrinet, Quadrics, Infiniband, MPI, Ethernet
- **Benefits:**
 - A network-portable binary UPC compiler now exists for x86, MIPS, future architectures supported by GCCUPC
 - Demonstrates that our runtime can be targeted by a binary compiler



GCCUPC: implementation



- **Primary obstacle: inline functions**
 - Needed in src-to-src for speed, abstraction layer.
 - But can't link against them from binary compiler
- **Current solution:**
 - GCCUPC generates performance-critical logic (ptr manipulation, MYTHREAD, etc.) as binary
 - Convert other inline functions into regular functions
- **Future: extra inlining pass**
 - Read our inline function definitions & generate binary code from them for shared accesses
 - Would allow GCCUPC to automatically get our platform-specific shared pointer representations



Topics



- Pthreaded runtime
- Support for GCCUPC (Intrepid)
- **C++/MPI Interoperability**
- Usability/stability improvements
- Future work



C++/Fortran/MPI Interoperability



- **Experiment came out of GCCUPC work**
 - Needed to publish an explicit initialization API
 - Made sure C++/MPI could use it, so we wouldn't have to change interface later.
- **Motivation: “2nd Front” for UPC acceptance**
 - Allow UPC to benefit existing C++/Fortran/MPI codes
 - Allow UPC code to use C++/Fortran/MPI libraries
 - Optimize critical sections of code
 - Communication, CPU overlap
 - Easier to implement certain algorithms
 - Easier to use than GASNet
 - Provide transparently in existing libraries (SuperLU)



C++/MPI Interoperability



- **Note: “This is not UPC++”**
 - We’re not supporting C++ constructs within UPC
 - C++/MPI can call UPC functions like regular C functions
 - UPC code can call C functions in C++/MPI code
 - UPC functions can return regular C pointers to local shared data, then convert them back to shared pointers to do communication
- **Status:**
 - Working in both directions: {C++/MPI} --> UPC, and vice versa
 - Tested with IBM xLC, Intel ecc, HP cxx, GNU g++, and their MPI versions.
 - Plan to ship in next release of Berkeley UPC (very very soon!)



UPC as a Library Language



- **Major limitation: can't share arbitrary data**
 - Can't share arbitrary global/stack/heap memory: must allocate shared data from UPC calls (`local_alloc`, etc.)
 - This problem would exist for UPC++, too.
- **“Shared everything” UPC**
 - Regular dynamic/heap memory: easy (hijack `malloc`)
 - Stack/global data: harder (but firehose allows)
 - Optional UPC extensions?
 - `UPC_SHARED_EVERYTHING`
 - Allow pointer casts from local --> shared.
- **Interoperability with MPI Communicators**
 - subgroup collectives, I/O
- **UPC libraries: static vs. dynamic threads**



Topics



- Pthreaded runtime
- Support for GCCUPC (Intrepid)
- C++/MPI Interoperability
- Usability/stability improvements
- Future work



Usability/Stability improvements



- **“Brainless” installation for new users**
 - Added remote translation over HTTP
 - Only need to download/install 5 MB runtime
 - Almost all networks are now autodetected
 - `configure; make; make install`
 - Can install Berkeley UPC in ~5 minutes



Usability/Stability improvements



- **Nightly build of runtime on many configurations:**

| | | |
|------------------|-----------------|--------------------|
| Linux | x86/IA64 | GM/VAPI |
| Tru64 | Alpha | Elan/MPI |
| AIX | Power 3 | LAPI/MPI |
| OS X | Power 5 | IB/MPI |
| SGI Altix | IA64 | pthread/MPI |
| T3E | Alpha | MPI |

- **Test suite now contains 250+ test cases**
 - works with IBM, Quadrics, PBS batch systems
 - **Nightly tests: 20 configurations, including all network types (both single/multi-threads, optimized/debug)**



Topics



- **Pthreaded runtime**
- **Support for GCCUPC (Intrepid)**
- **C++/MPI Interoperability**
- **Usability/stability improvements**
- **Future work**



Future work



- **System V shared memory support**
- **GCCUPC inline pass support**
- **Caching remote shared accesses**
- **“Shared everything” UPC**
- **Totalview debugger support**