Berkeley UPC

http://upc.lbl.gov

PGAS

Overview

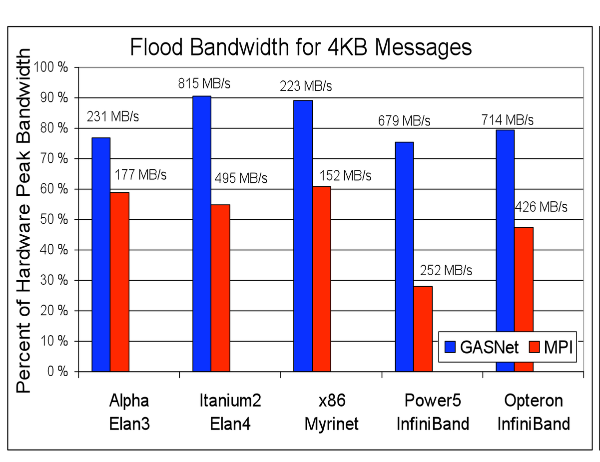
- A portable and high-performance UPC implementation, compliant with UPC 1.2 spec
- Features:
  - High performance UPC Collectives
  - Extensions for performance and programmability
    - Non-blocking memcpy functions
    - Semaphores and signaling put
    - Value-based collectives
    - Atomic memory operations
    - Hierarchical layout query
  - Compiler and runtime optimizations for application scalability
- Open source software (Windows/Mac/UNIX), installation DVD available

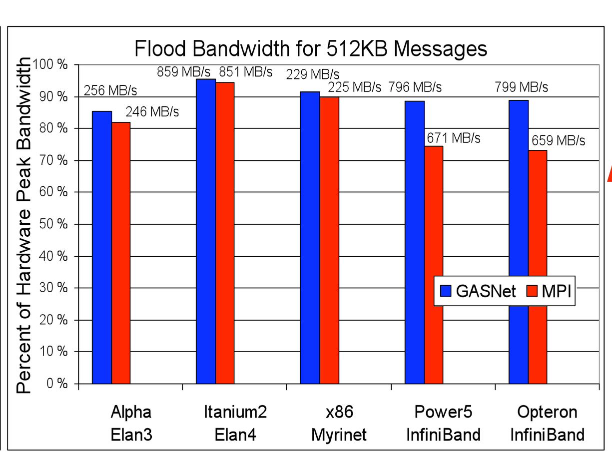
Portable Design

- Layered design, platform-independent code gen
- Supports wide range of SMPs, clusters and MPPs
  - x86, Itanium, Opteron, Athlon, Alpha, PowerPC, MIPS, PA-RISC, SPARC, T3E, X1, SX-6, XT3, Blue Gene, …
  - Linux, FreeBSD, NetBSD, Tru64, AIX, IRIX, HPUX, Solaris, MS Windows, Mac OS X, Unicos, SuperUX, …
  - Pthreads, Unix SysV, Myrinet, Quadrics Elan 3/4, InfiniBand, IBM LAPI, Dolphin SCI, MPI, Ethernet, Cray X1 / SGI Altix shmem, Cray XT Portals, IBM BG/P DCMF (new: see poster)

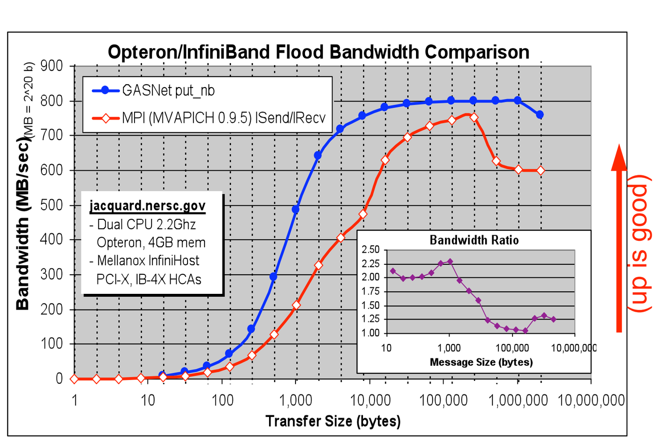
BUPC Runtime + GASNet

- Well-documented runtime interface, multiple UPC compilers (Berkeley UPC and Intrepid GCC/UPC)
- Debugging and tracing support
  - Performance Instrumentation Support (GASP)
  - Supports Parallel Performance Wizard (PPW)
  - Detailed communication tracing support
  - Etnus TotalView debugger support
- Interoperability with other programming env:
  - UPC calls to/from C, C++, Fortran, MPI
- Berkeley GASNet used for communication:
  - Performance from inline functions, macros, and network-specific implementations
- Optimized Collective ops
- High-performance communication
  - Consistently matches or outperforms MPI
  - One-sided, lightweight semantics

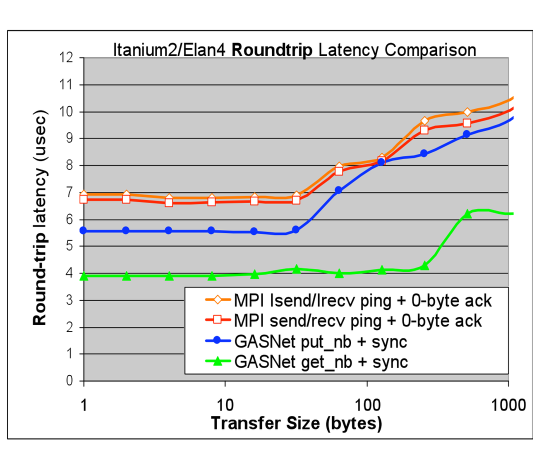




(up is good)



(up is good)



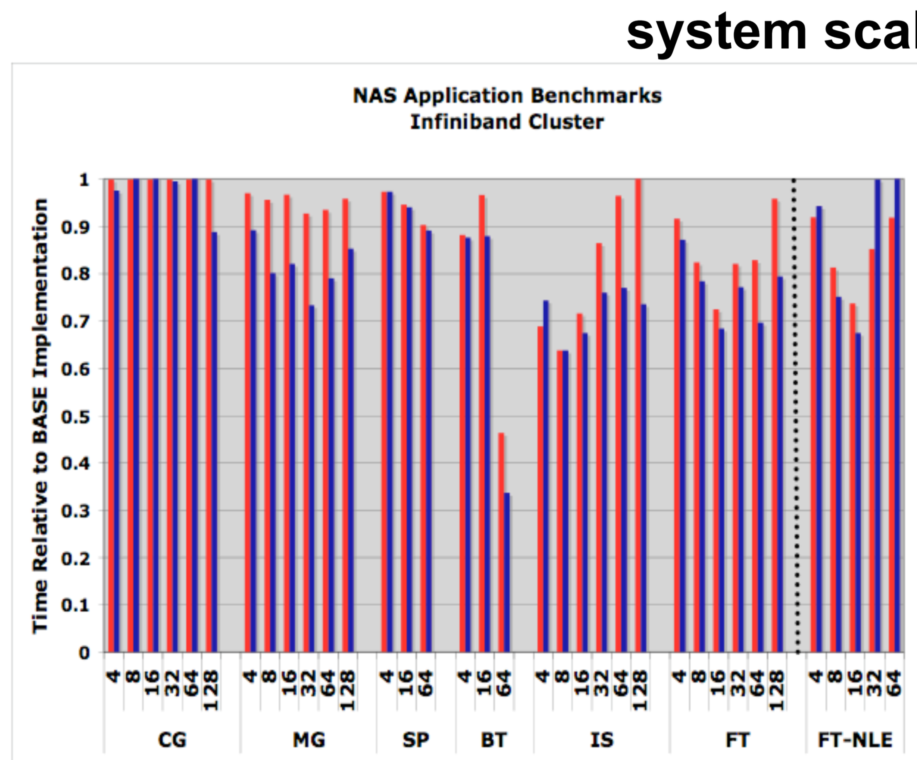
(down is good)

UPC-to-C Translator

- Source-to-source translator, based on Open64
- Enhances programmer productivity through static and dynamic optimizations: compiler, runtime, communication libraries

Performance Portability: System, Scale, Load

- Compile time message vectorization and strip-mining
- Runtime Analysis: communication instantiated at runtime based on system specific performance models
- Performance models designed to take system scale and load into account



Communication dynamically instantiated using either Put/Get or VIS calls

Overall improved application scalability and programmer productivity

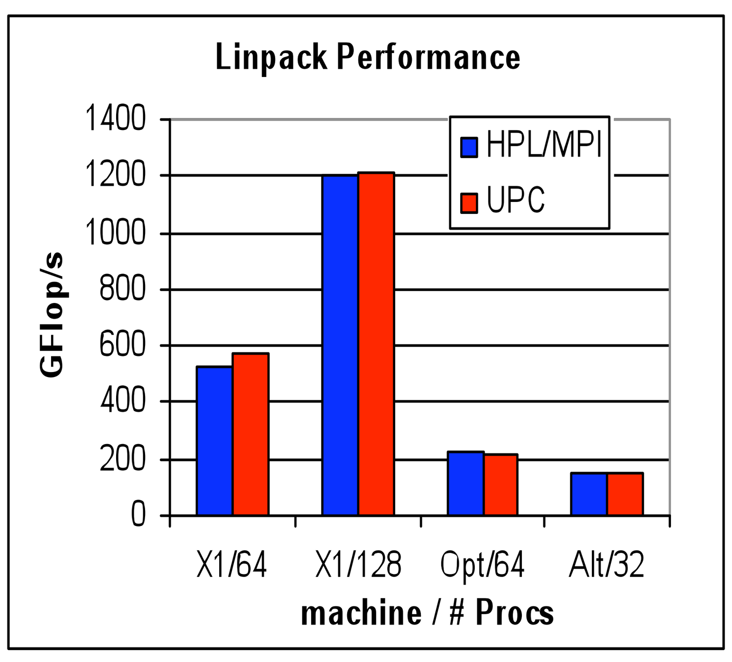
Multithreading for Latency Hiding

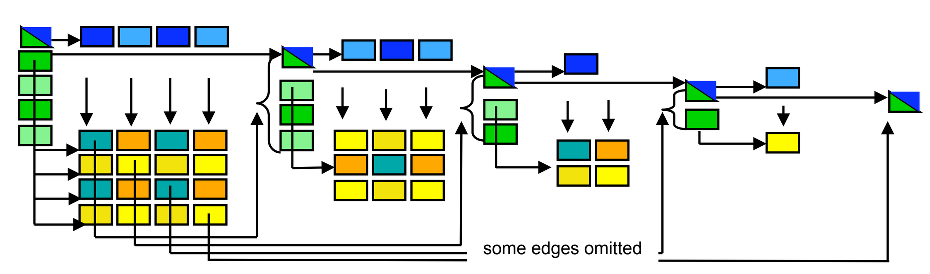
For distributed memory or heterogeneous architectures one needs:

- Locality and Load Balance
- Remote synchronization for dependencies
- Latency Tolerance

Case Study - Linpack

- 2d block cyclic decomposition as in ScaLAPACK
- Cooperative multi-threading to mask dependences
- Non-blocking (remote get) transfers to mask latency
- Memory-constrained lookahead compared to none in ScaLAPACK, fixed parameter in MPI/HPL
- Application-level scheduling to prioritize critical path



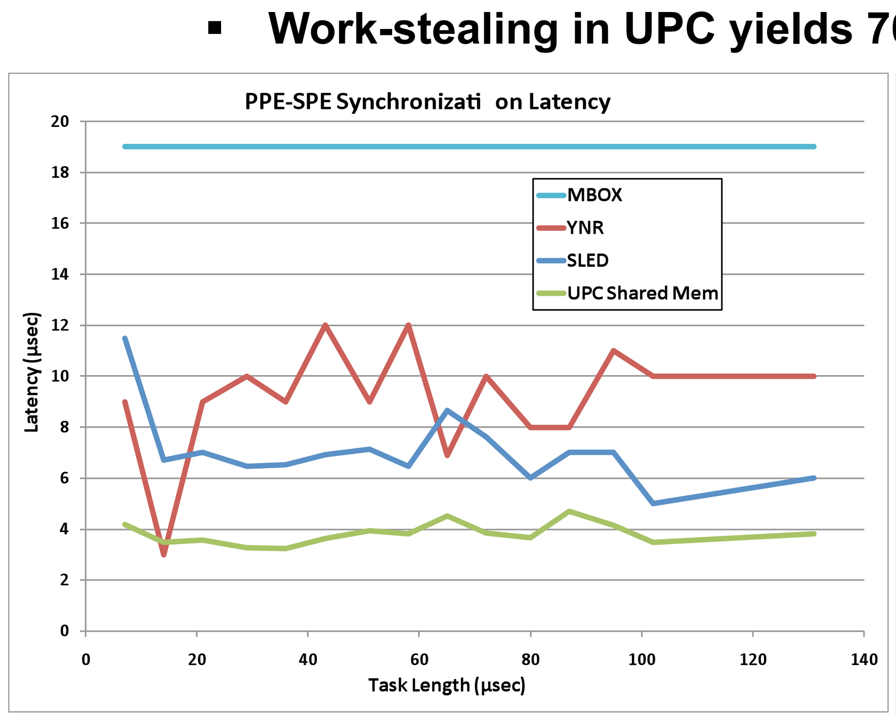


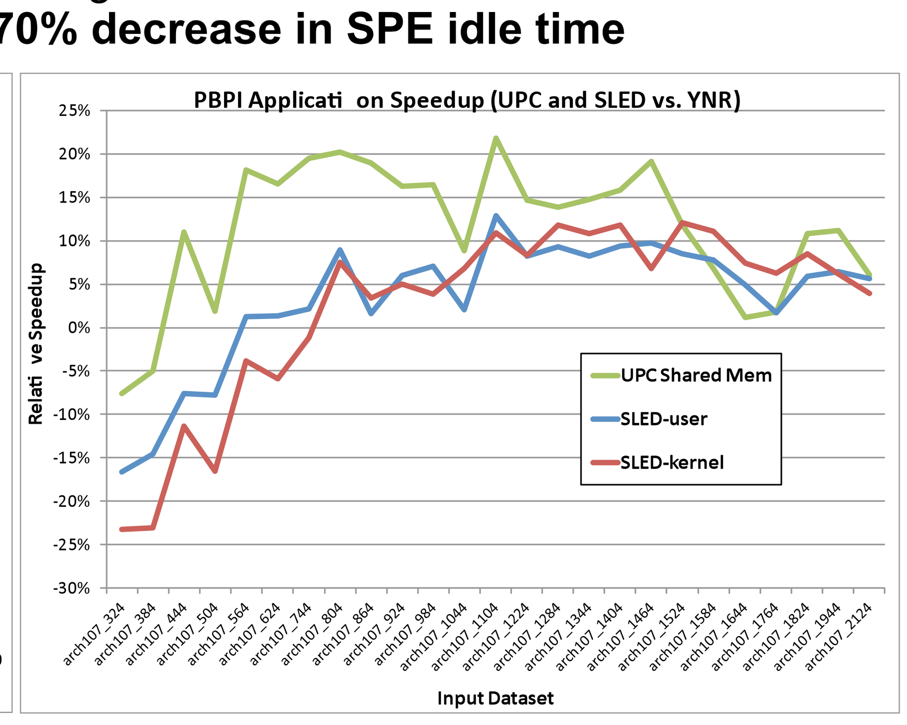
Other results:
Itanium 2/Elan 4.1 – 2.25 TFlop/s, 78.5% of peak on 512p
1p Itanium 2 1.5 GHz – 91.8% of peak
1p Opteron 2.2GHz – 81.9% of peak

Case Study – Cell BE (Sony PS3)

- Disjoint hardware hierarchies with different degrees of parallelism
- Bioinformatics applications: PBPI and RAXML
- Oversubscription (multi-threading) masks dependences and increases utilization
- Cooperative scheduling to minimize SPE idle time
- Asynchronous PPE-SPE interaction
- Compares performance for 4 algorithms
  - MBOX: SPE Mailboxes from Cell SDK
  - YNR: "Yield if Not Ready"
  - SLED: "SLack-minimizer Event-Driven"
  - UPC Shared Mem: work stealing in UPC
- Work-stealing in UPC yields 70% decrease in SPE idle time

(down is good)





(up is good)

BERKELEY LAB

BERKELEY UPC UNIFIED PARALLEL C