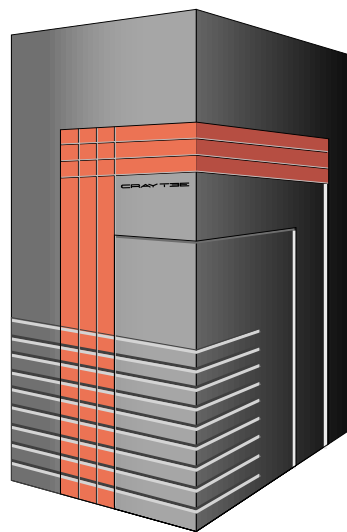


Performance Lessons from the CRAY T3E System



Jeff Brooks, Benchmarking Dept
SGI
jpb@sgi.com

A 50 year-old idea..

Hierarchy of Memories...

- Ideally one would desire an indefinitely large memory capacity such that any particular .. word would be immediately available–i.e. in a time which is somewhat or considerably shorter than the operation time of a fast electronic multiply ... It does not seem possible physically to achieve such a capacity. *We are therefore forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible.**

**A Burks, HH Goldstine and J von Neumann. Preliminary Discussion of an Electronic Computing Instrument, part 1 vol. 1, Institute for Advanced Study, Princeton, June 1946 (2nd edition September 1947). Reprinted in B Randell (ed.).*

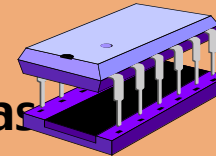
Question: How many levels of memory (visible to the programmer) exist in the Cray T3E?

Technological Scaling Drivers

- **Density (on-chip performance):** Scales at “Moore’s Law” rate: 2x every ~18 months.
- **Connectivity (between chip performance):** Scales at *slower* rate: 2x every ~26–30 months.

Example: 1979 → 1999 DRAM:

- 16,000X density increase
- 640X uniform access BW increase
- 500X random access BW increase



Net Result: The need to recognize and exploit memory hierarchies will grow *increasingly important* for scalable algorithms, programming languages & compilers.

Kinds of Locality

- **Spatial Locality**

- Cache works fine with effective VL = cache width
- Vectors probably a “better” technology

- **Temporal Locality**

- Current vector architecture can't take advantage of this very well.
- Excellent structure for data caches.

- **No Locality**

- Need some latency hiding technique
- Need true memory bandwidth

```
do i = 1, n
    a(i) = b(i) + c(i)
enddo
```

```
do i = 1, n
    a(i) = x(i) + x(i-1)
enddo
-or-
call sub1(x,y)
call sub2(x,y)
```

```
do i = 1, n
    b(j,i) = a(index(i))
enddo
```


Cache Resident Computation (programming to the fastest memory level...)

- The dot product of two complex vectors **cx** and **cy**:

$$c\alpha = cx(1)*cy(1) + cx(2)*cy(2) + \dots + cx(n)*cy(n)$$

- Running in-cache, and unrolled by 8, and assembly version of this loop was observed to run a **1180 Mflops** out of a possible 1200 on the CRAY T3E1200.
- Your results will probably be less...

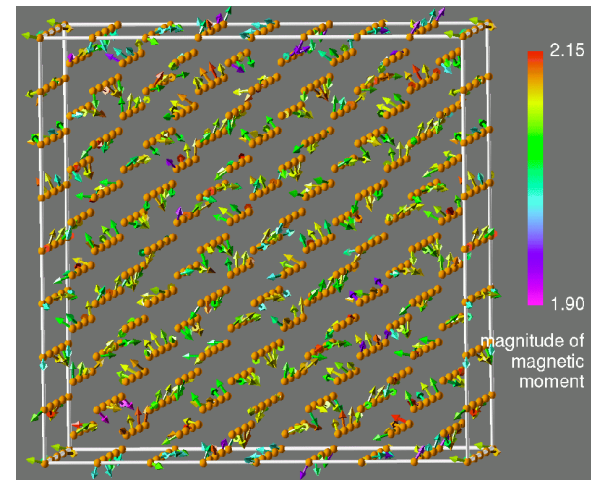
Cache Resident Full Code Example:

LSMS: Locally self-consistent multiple scattering method. Metallic magnetism simulation for understanding thin film disc drive read heads, magnets used in motors and power generation.

1.02 Teraflops on CRAY T3E1200 with 1480 processors

1998 Gordon Bell Prize winner !

Makes heavy use of CGEMM.



B. Ujfalussy, Xindong Wang, Xiaoguang Zhang, D. M. C. Nicholson,

W. A. Shelton, and G. M. Stocks,

Oak Ridge National Laboratory, Oak Ridge, TN 37831,

- A. Canning,

NERSC, Lawrence Berkeley National Laboratory, Berkeley, CA 94720,

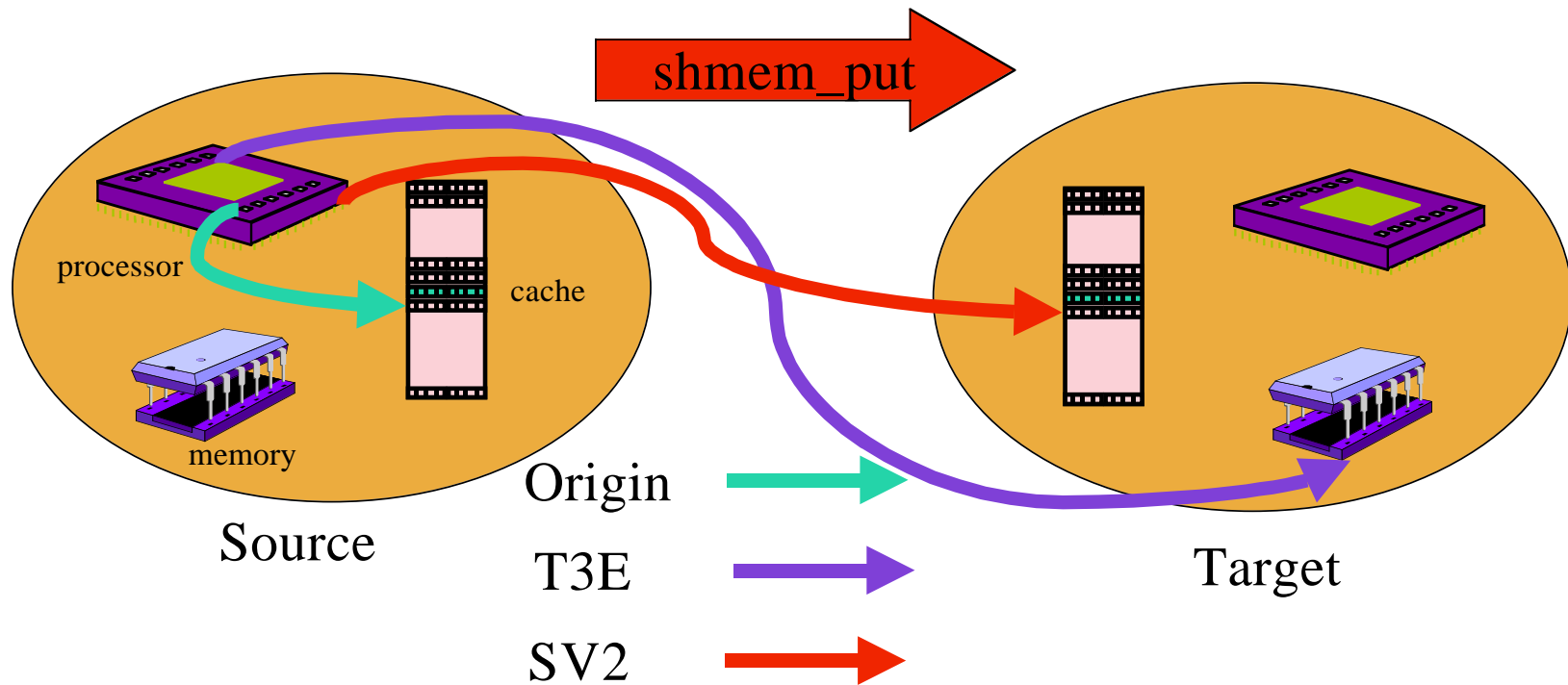
- Yang Wang,

Pittsburgh Supercomputing Center, Pittsburgh, PA 15213,

- B. L. Gyorffy,

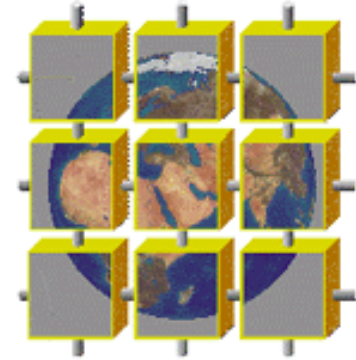
H. H. Wills Physics Laboratory, University of Bristol, Bristol, UK.

shmem_put: Where does the data go?



Distributed memory programming models work better if you *don't* cache remote data.

NASA HPC Earth and Space Sciences Project

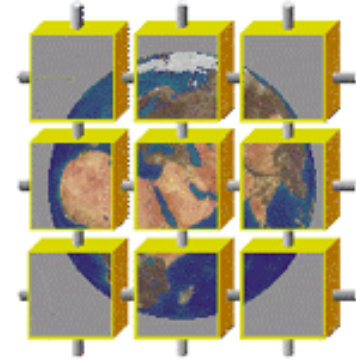


- **MHDPPM: PPM with magnetohydrodynamics**
196 GF on 1024 T3E 1200. Vectorizable
- **MHD: A fully pseudo-spectral code in 32-bit precision.**
160 GF on 1024 T3E 1200. Vectorizable
- **HPS: Hybrid pseudo-spectral finite difference code.**
167 GF on 1024 T3E 1200. Vectorizable
- **CACTUS: Fully relativistic hydro code for simulating neutron star collisions**
142 GF on 1024 T3E1200. Vectorizable.
- **DYNAMO: Pseudo-spectral code for solving the geodynamo problem.**
120 GF on 512 T3E1200. Vectorizable.
- **MGFLO: Finite Element Model. Carey Team from UT Austin.**
147 GF on 1024 T3E1200. Vectorizable.
- **BATS-R-US: Block-Adaptive-Tree Solar-wind Roe-type Upwind Scheme.**
212 GF on 1024 T3E 1200. Vectorizable.

T3E Optimization Techniques

- **Think Distributed Data & Algorithm!**
 - Distributed data programming models have a track record of high performance.
 - Use SHMEM or co-arrays for ease of programming.
 - If portability is required, use MPI for most communication and SHMEM only where low-latency and very high-performance is required.
- **Think Vector!**
 - Use “fat” vector loops if you can run out of DCACHE.
 - Use “skinny” vector loops if you are running out of stream buffers.
 - Use E-registers for structures without spatial locality.

NASA HPC Earth and Space Sciences Project



- **Optimization techniques & Observations:**
 - Some codes chunked up for DCACHE to exploit temporal locality, but still vectorize perfectly (PPM). (One key loop for CACTUS has 50 state variables and over 3000 floating point operations are performed. It vectorizes, but has wonderful temporal locality)
 - Codes that don't fit cleanly in DCACHE are optimized for the T3E's stream buffers (TERRA uses streams very successfully).
 - MGFLO exploits the BLAS library matrix-vector multiply (SGEMV) which, of course, vectorizes well and exploits streams on the T3E.
 - HPS & MH3D: Biggest concern is the global transpose of the data which uses cache-bypassing E-register facility for data motion.

System Balance: Operational Weather Codes



- Hirlam: 514x514x16 gridpoint model ran at 82.8 Gigaflops on 1024 processors.



- IFS: Spectral model from ECMWF ran at 95 Gigaflops on 1408 processors

	PEs	flops per word sent	Number of Msgs	Avg Message Length
Hirlam	1024	125	3000000	20
IFS	16	800	1500	40000

*Data from: Deborah Salmond, Alan Dickinson, Nis Gustafsson & Bob Carruthers.
Operational Numerical Weather Prediction Models on the Cray T3E.*

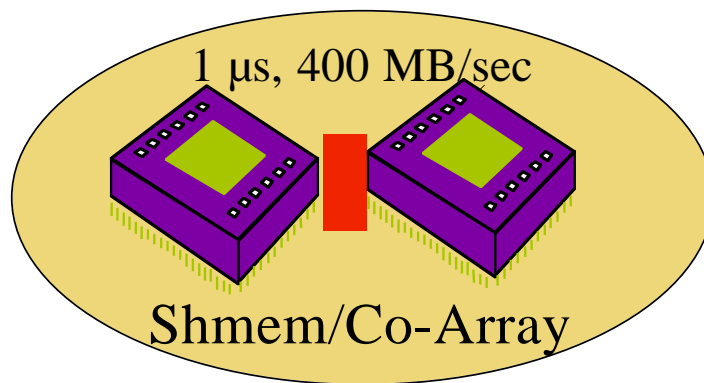
System Balance: Cray T3E 1200



Typical tuned single-
pe performance = 120
Mflops



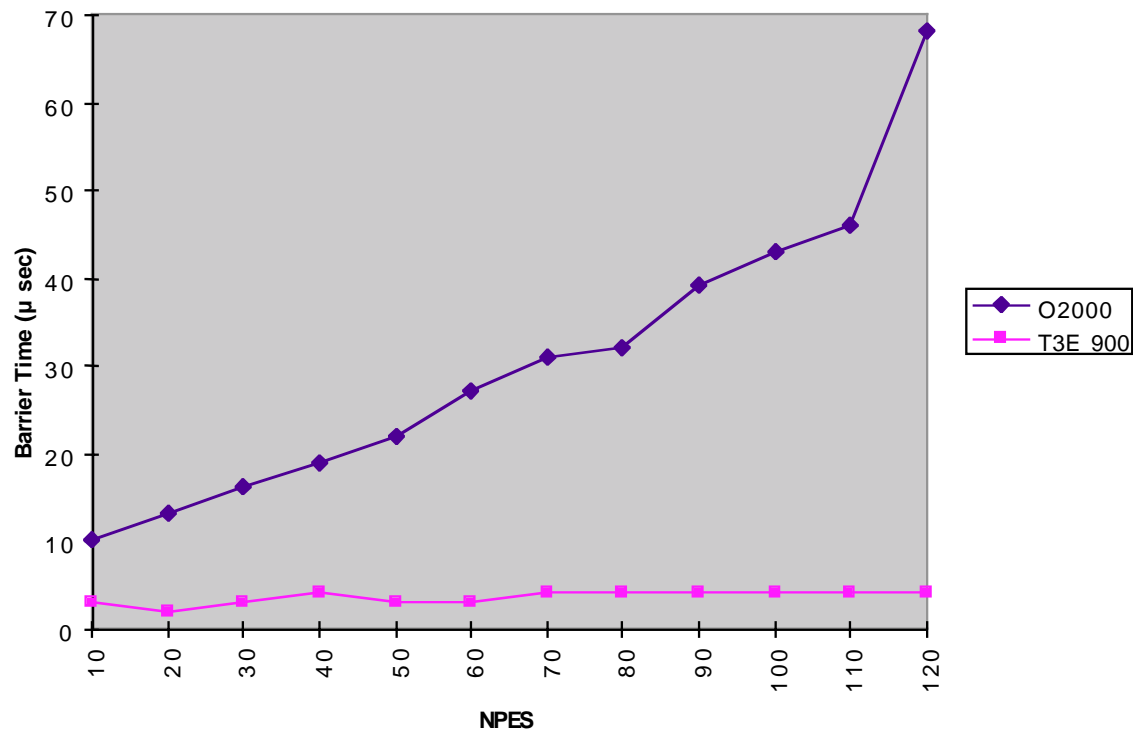
Good balance to highly
scale spectral model (IFS)



Good balance to highly scale
grid-point model (Hirlam) or
spectral model

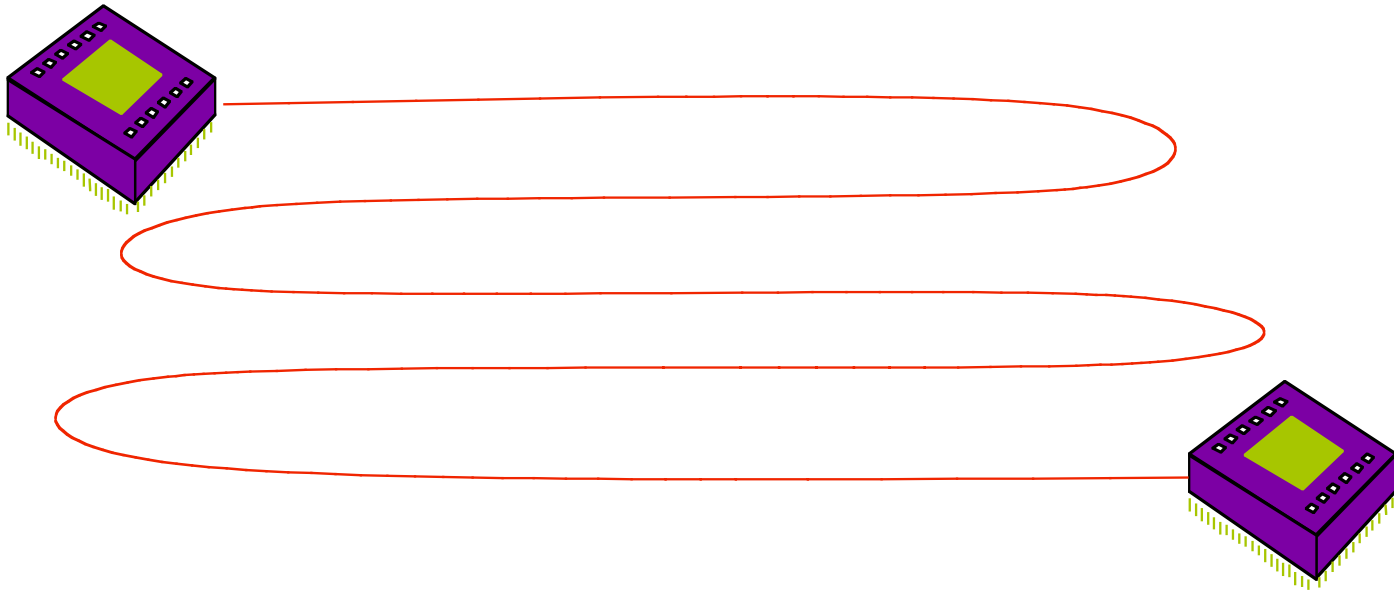
Origin 2000 and T3E Barrier Performance

- The CRAY T3E has a fast hardware barrier facility. The O2000 implementation is done via memory operations and software.



Fast barriers are essential for low-latency programming styles.

Question: What Kind of System is This?



sgi

CRAY
RESEARCH

Answer: Avalon Beowulf System




Avalon is a 140 Alpha Beowulf cluster constructed at Los Alamos with fast ethernet switches.

MPI Latency $\sim 100 \mu\text{s}$

MPI Bandwidth $\sim 10 \text{ MB/sec}$

This is enough capability for some problems including:

Linpack:	47 Gflops
SPaSM (molecular dynamics):	12.8 Gflops
NPB BT:	2.2 Gflops
NPB SP:	1.0 Gflops
NPB LU:	3.5 Gflops
NPB MG:	2.1 Gflops



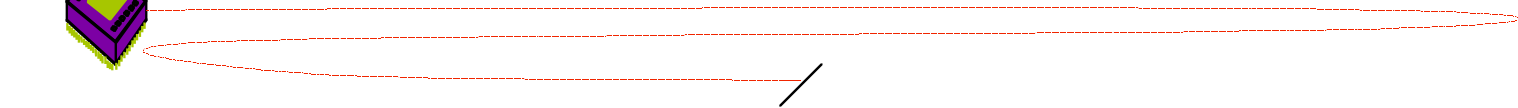
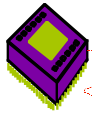
$(\text{Probably})^{1024} = \text{Probably Not}$

So probably is not good enough...

Probably...

- **Will all processors give me equivalent performance?**
 - Memory “free” of competing jobs taking bandwidth or space?
 - Does the scheduler work?
 - Page layout the same on each processor?
 - Can you guarantee that I *never* take a virtual memory page miss on any processor?
 - Processors “free” of competing processes or system daemons?
- **Will the machine stay up for my job?**
 - Are all parts 99.99999% reliable?
- **Will I get the “right” answer?**
 - Memory/register/cache errors detected/corrected?
 - Does the message passing fabric have error detection/correction?

Question: What Kind of System is This?



15 Kilometers



sgi

CRAY
RESEARCH

Answer: Internet SETI@HOME Screen Savers

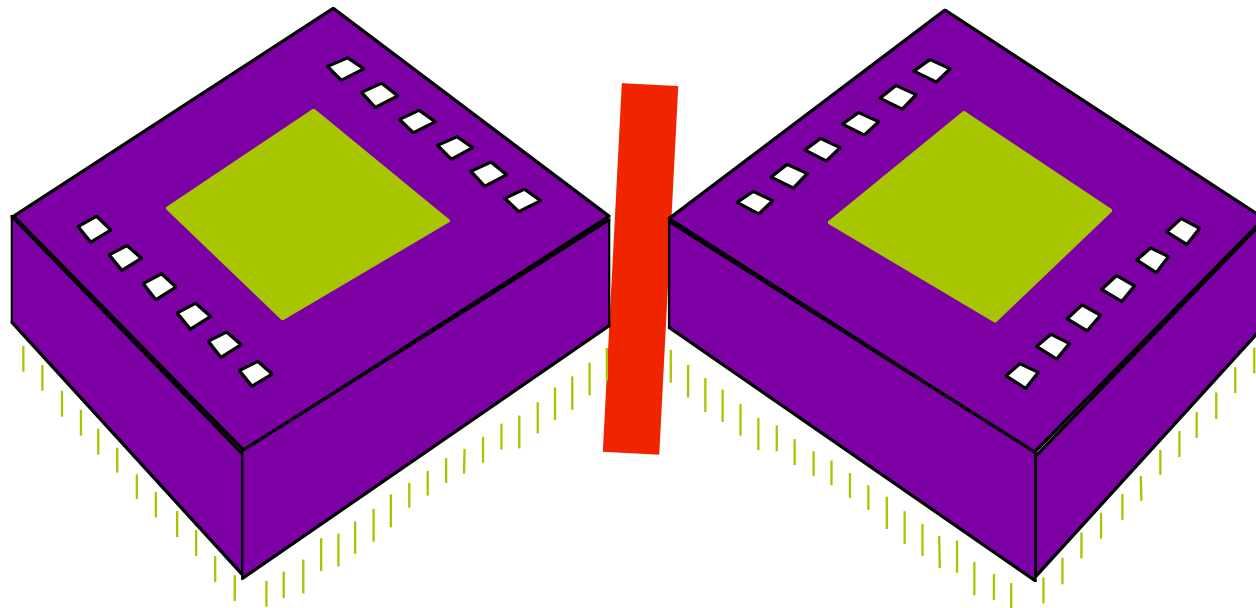


Last updated: Fri Sep 3 23:00:11 1999 UTC

Last 24-hour statistics:

CPU's	481070
CPU Time	1318 years
Floating Point Operations	11.8 TeraFLOP/sec

Cray T3E follow on...



- More than an order of magnitude increase in sustained performance per processor.
- More than an order of magnitude increase in communication bandwidth.
- ~1 μ s remote memory access

T3E → SV2: What did we learn?

T3E System

Direct addressable memory over entire system allowing for low-latency, one-way communication models. E-Registers required to address remote memory. OS is not involved in data motion.

“Scalable” memory translation (remote translation mechanism)

10 to 1 performance ratio between cache and memory

Fast barrier capability

SV2 System

Directly addressable memory over entire system via normal processor loads & stores. OS not involved in data motion.

Same

3 to 1 performance ratio between cache and memory.

Same

T3E → SV2: What did we learn?

T3E System

Mechanism to address memory efficiently with strided or random address patterns. (E-registers)

Programmer can bypass cache when appropriate.

Stream buffers to hide memory latency and improve bandwidth for spatial-locality constructs (i.e. stride-1 loops).

No SMP capability

SV2 System

Better mechanism to address memory efficiently with strided or random address patterns (vector registers).

Same

Vector registers to hide memory latency.

SMP capability. Size of Cache domain will be smaller than the whole machine and *it can be turned off!*

T3E -> SV2: What did we learn?

T3E System

Bufferless MPI, SHMEM, Co-Array FORTRAN supported.

Space-sharing capability in OS

Good global resource scheduling.

Checkpoint/restart, NQE, redundant capability, other UNICOS mk functionality.

Supercomputer packaging: 100s of Gigafllops sustained in a reasonable* footprint.

SV2 System

Bufferless MPI, SHMEM, Co-Array FORTRAN and OpenMP supported.

Same

Same

Same.

Supercomputing packaging. Multiple Terafllops sustained in a reasonable footprint.

*what is reasonable?: You won't have to build a new computer room

T3E → SV2 What did we learn?

T3E System

Best single-processor performance obtained from dependency-free loops running from cache or stream buffers.

Best parallel performance obtained from a well thought out algorithm written in message passing with low-latency SHMEM library used where appropriate.

SV2 System

Best single processor performance obtained from dependency-free loops running from cache or memory via vector loads and stores.

Best parallel performance obtained from a well thought out algorithm written in message passing with low-latency SHMEM library used where appropriate.