

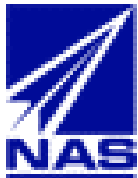
Comparing the OpenMP, MPI, and Hybrid Programming Paradigm on an SMP Cluster

G. Jost*, H. Jin*, D. an Mey**, F. Hatay***

**NASA Ames Research Center*

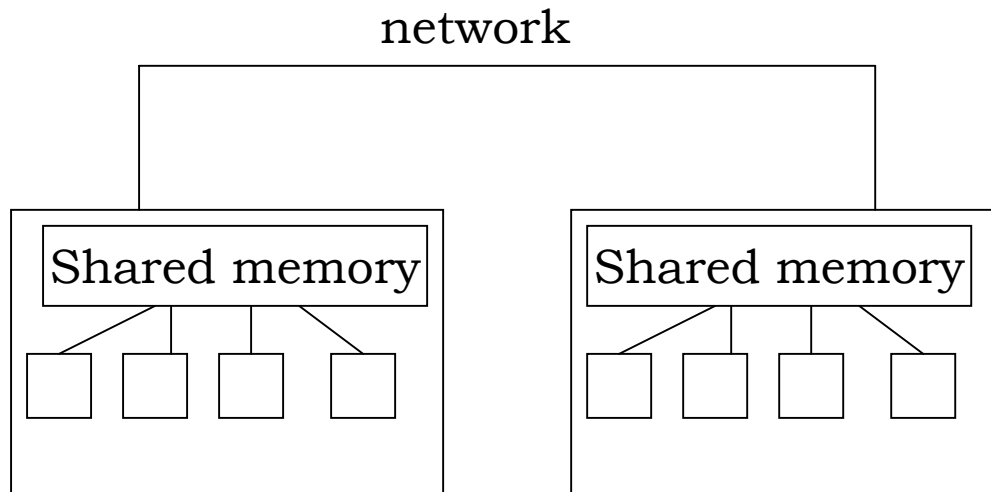
***Center for Computing and Communication,
University of Aachen*

****Sun Microsystems*



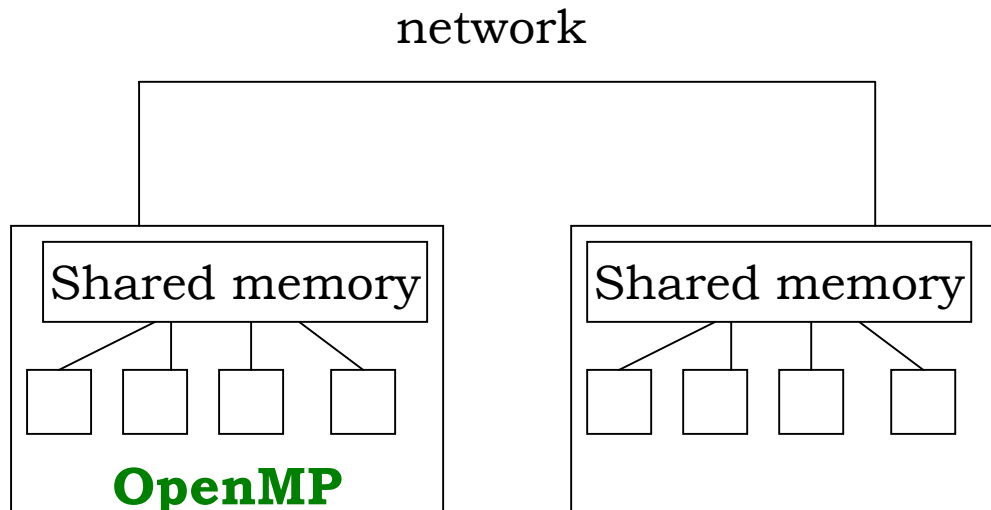
SMP Cluster

- **Cluster of Symmetric Multi-Processor (SMP) nodes**
- **Mix of shared and distributed Memory Architecture**
- **Programming Paradigms:**
 - ▶ OpenMP within one SMP node
 - ▶ All message-passing
 - ▶ Hybrid programming:
 - Shared memory with one SMP node
 - Message-passing between SMP nodes



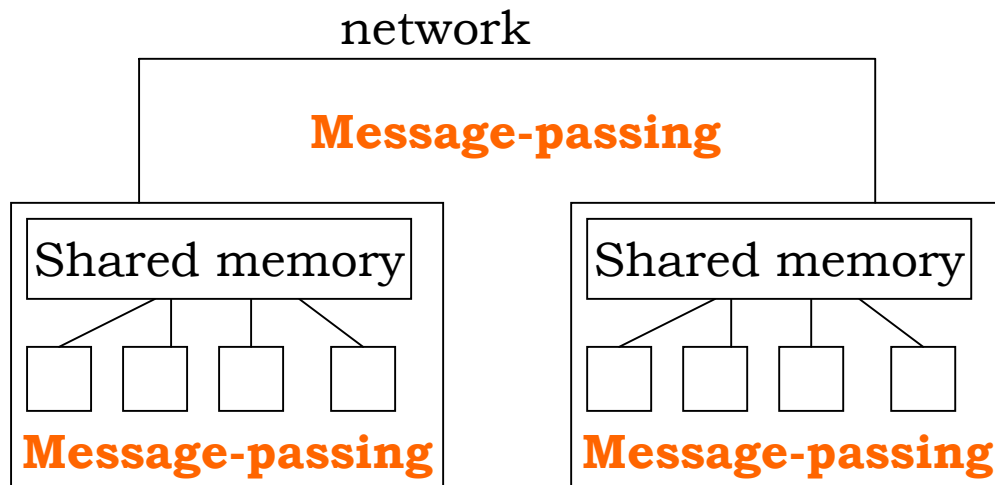
SMP Cluster

- **Cluster of Symmetric Multi-Processor (SMP) nodes**
- **Mix of shared and distributed Memory Architecture**
- **Programming paradigms:**
 - ▶ **OpenMP within one SMP node**
 - ▶ All message-passing
 - ▶ Hybrid programming:
 - Shared memory with one SMP node
 - Message-passing between SMP nodes



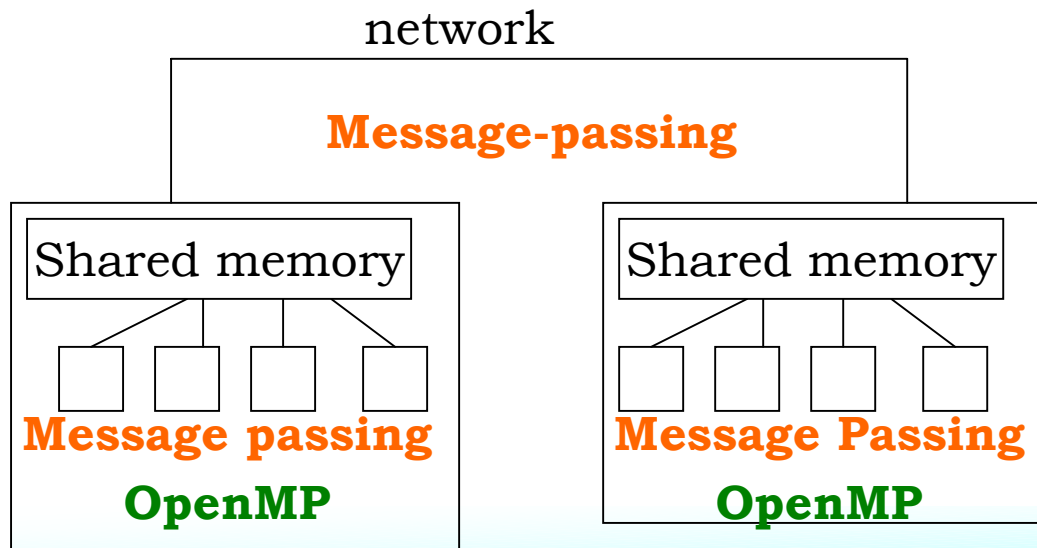
SMP Cluster

- **Cluster of Symmetric Multi-Processor (SMP) nodes**
- **Mix of shared and distributed memory architecture**
- **Programming paradigms:**
 - ▶ OpenMP within one SMP node
 - ▶ **All message-passing**
 - ▶ Hybrid programming:
 - Shared memory with one SMP node
 - Message-passing between SMP nodes



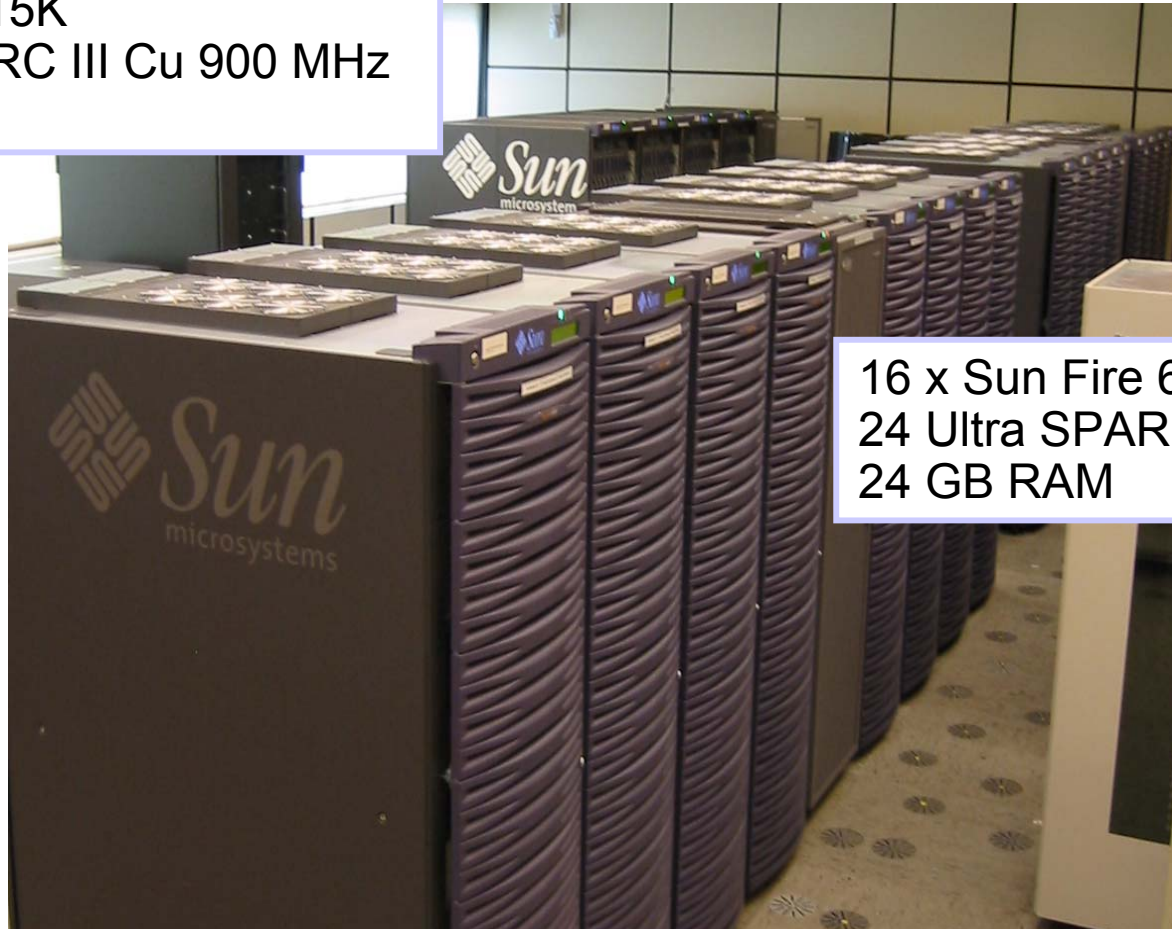
SMP Cluster

- **Cluster of Symmetric Multi-Processor (SMP) nodes**
- **Mix of shared and distributed memory architecture**
- **Programming paradigms:**
 - ▶ OpenMP within one SMP node
 - ▶ All message-passing
 - ▶ **Hybrid programming:**
 - **OpenMP within one SMP node**
 - **Message-passing between SMP nodes**



Sun Fire SMP Cluster

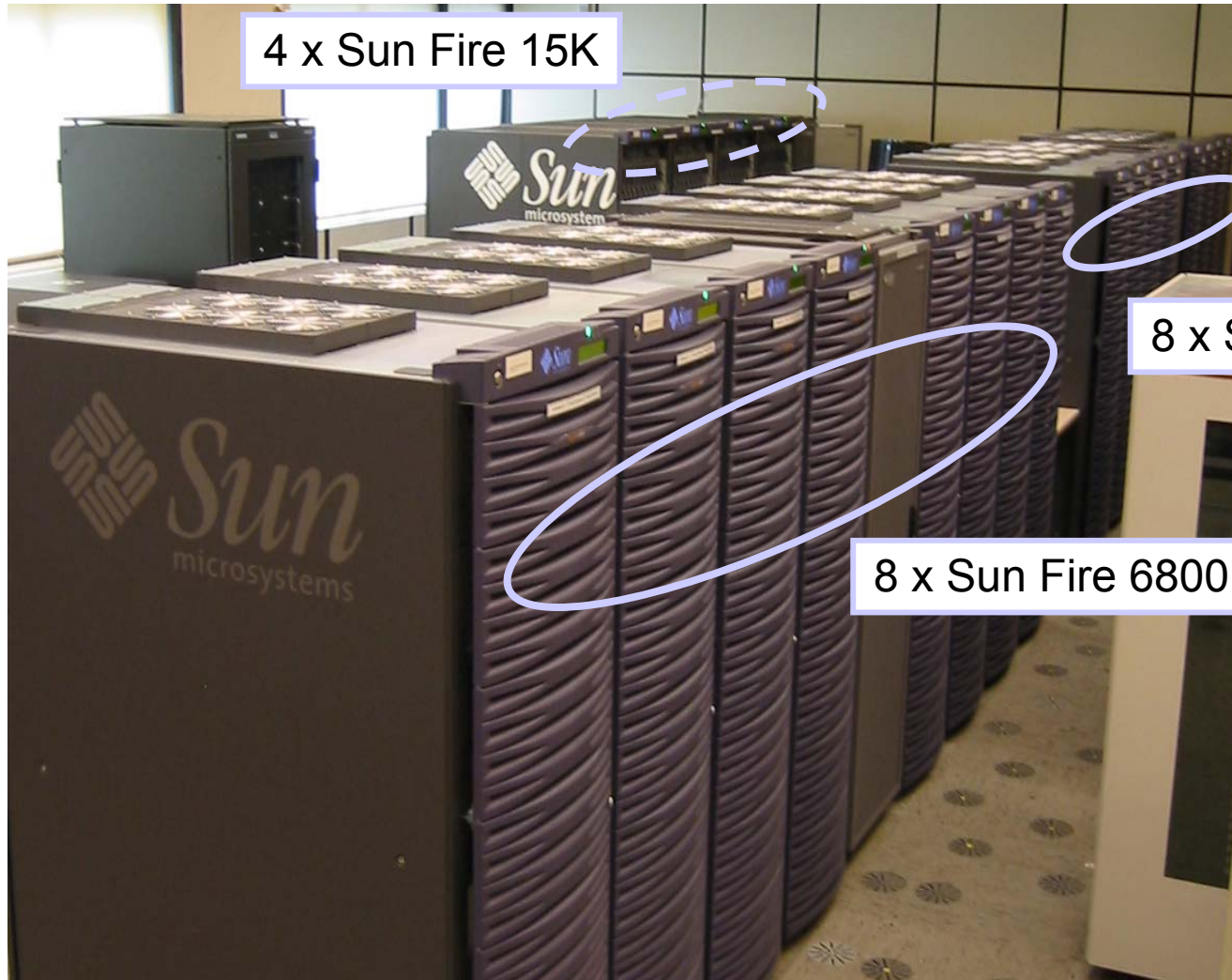
4 x Sun Fire 15K
72 Ultra SPARC III Cu 900 MHz
144 GB RAM



16 x Sun Fire 6800
24 Ultra SPARC III Cu 900 MHz
24 GB RAM

- **Located at the Computing Center of the University of Aachen (Germany)**

Sun Fire Link Configuration



Hardware Details

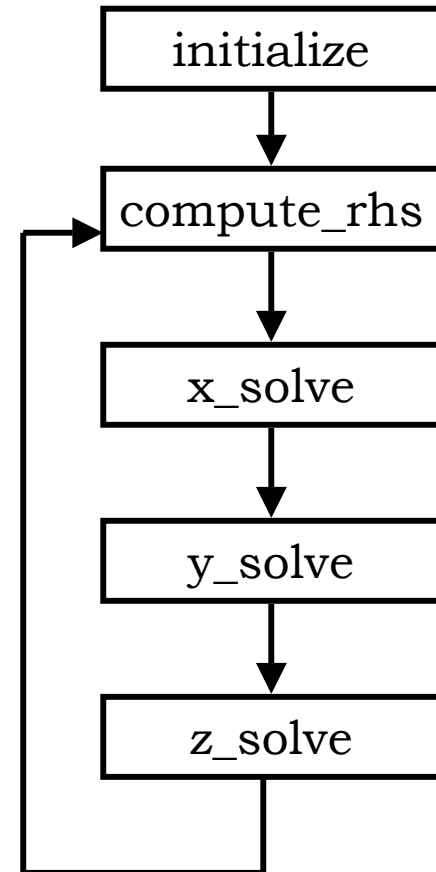
- **UltraSPARC-III Cu processors**
 - ▶ Superscalar 64-bit processor
 - ▶ 900 MHz
 - ▶ L1 cache (on chip) 64KB data and 32KB instructions
 - ▶ L2 cache (off chip) 8 MB for data and instructions
- **Sun Fire 6800 node:**
 - ▶ 24 UltraSPARC-III 900 MHz CPU
 - ▶ 24 GB of shared main memory
 - ▶ Flat memory system:
 - approx. 270 ns latency
 - 9.6 GB/s bandwidth
- **Sun Fire 15K node:**
 - ▶ 72 UltraSPARC-III 900 MHz CPU
 - ▶ 144 GB of shared main memory
 - ▶ NUMA memory system:
 - Latency 270 ns onboard to 600 ns off board
 - Bandwidth 173GB/s on board to 43 GB (worst case)

Testbed Configurations

- **Gigabit Ethernet (GE) MPI:**
 - ▶ 100 us latency
 - ▶ 100 MB/s bandwidth
- **Sun Fire Link (SFL) MPI:**
 - ▶ 4 us latency
 - ▶ 2GB/s bandwidth
- **4 Sun Fire 6800 nodes connected with GE**
 - ▶ 96 (4x24) CPUs total
- **4 Sun Fire 6800 nodes connected with SFL**
 - ▶ 96 (4x24) CPUs total
- **1 Sun Fire 15K node**
 - ▶ 72 (1x72) CPUs total

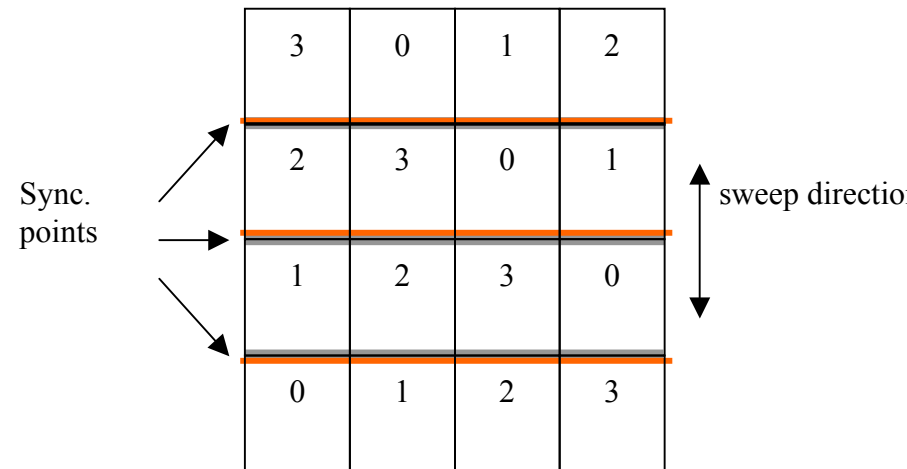
The NAS Parallel Benchmark BT

- **Simulated CFD application**
- **Uses ADI method to solve Navier-Stokes equations in 3D**
- **Decouples the three spatial dimensions**
- **Solves a tridiagonal system of equation in each dimension**
- **Four different implementations are used for this study:**
 - ▶ One pure MPI version
 - ▶ One pure OpenMP version
 - ▶ Two different hybrid MPI/Open versions



BT MPI

- **Message passing implementation based on MPI (as in NPB2.3)**
- **Domain decomposition employing a 3-D multi-partition scheme**
- **For each sweep directions all processes can start work in parallel**
- **Before each iteration:**
 - ▶ Exchange of boundary data
- **Within each sweep:**
 - ▶ Sends partially updated results to neighbors



BT OMP

- **Loop level parallelism based on OpenMP (as in NPB3.0)**
- **Place OpenMP directives on outermost loops in time consuming**
- **One dimensional parallelism**
- **Different spatial dimensions parallelized in different routines**

BT OMP code segment

```
!$omp parallel do
  do k= 1, nz
    do j= 1, ny
      do i=1,nx
        .. = u(i,j,k-1)
           + u(i,j,k+1)
      enddo
    enddo
  enddo
```

BT Hybrid V1

- **MPI/OpenMP:**

- ▶ 1D data partition in z-dimension (**k-loop**)
- ▶ OpenMP directives in y-dimension (**j-loop**)

Routine `y_solve`

```
!$omp parallel
do k=k_low,k_high
    synchronize neighbor threads
!$omp do
do j=1,ny
do i=1,nx
    rhs(i,j,k) = rhs(i,j-1,k)
                + ...
    enddo
enddo
synchronize neighbor threads
enddo
!$omp end parallel
```

- ▶ *data dependence in y-dimension*
- ▶ *pipelining of thread execution*

Routine `z_solve`

```
!$omp parallel do
do j=1,ny
    call receive
do k=k_low,k_high
do i=1,nx
    rhs(i,j,k) = rhs(i,j,k-1)
                + ...
    enddo
enddo
call send
enddo
```

- ▶ *data partitioned in K*
- ▶ *communication within parallel region*

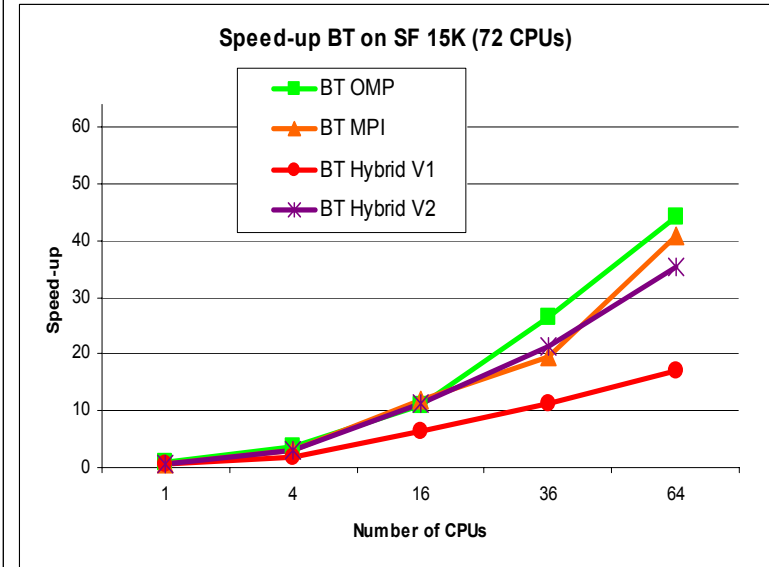
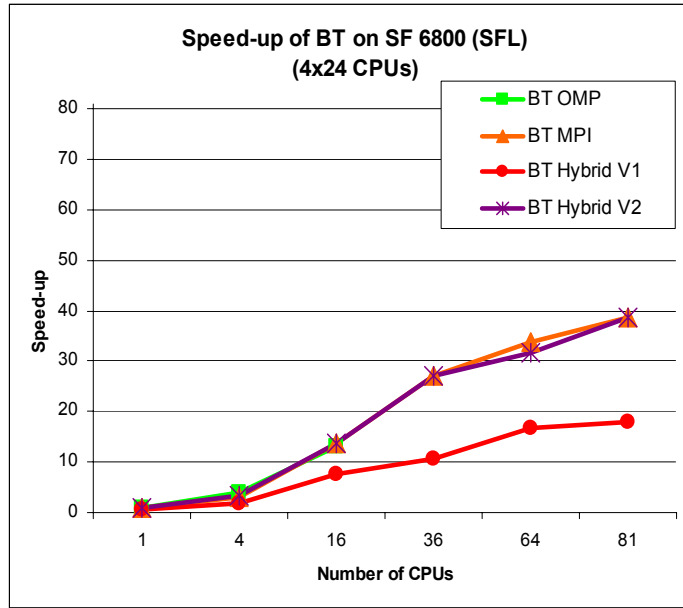
BT Hybrid V2

- **MPI/OpenMP:**
 - ▶ Employ 3-D multi-partition scheme from BT MPI
 - ▶ Place OpenMP directives on outermost loops in time consuming routines
 - Hybrid V2 without OpenMP
⇔ BT MPI
- **Differences to BT Hybrid V1:**
 - ▶ Data decomposition in 3 dimensions
 - ▶ MPI and OpenMP employed on the same dimension
 - ▶ All communication outside of parallel regions

Routine z_solve

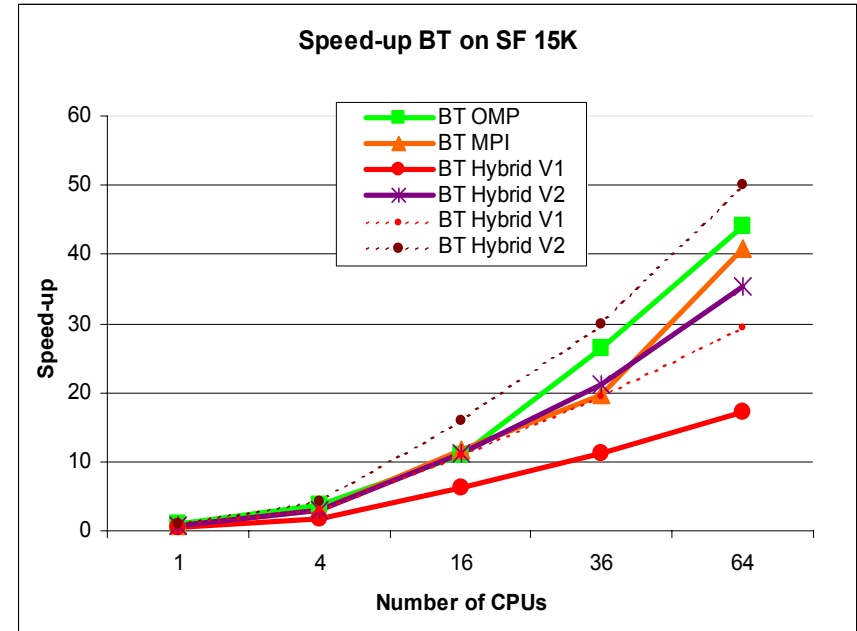
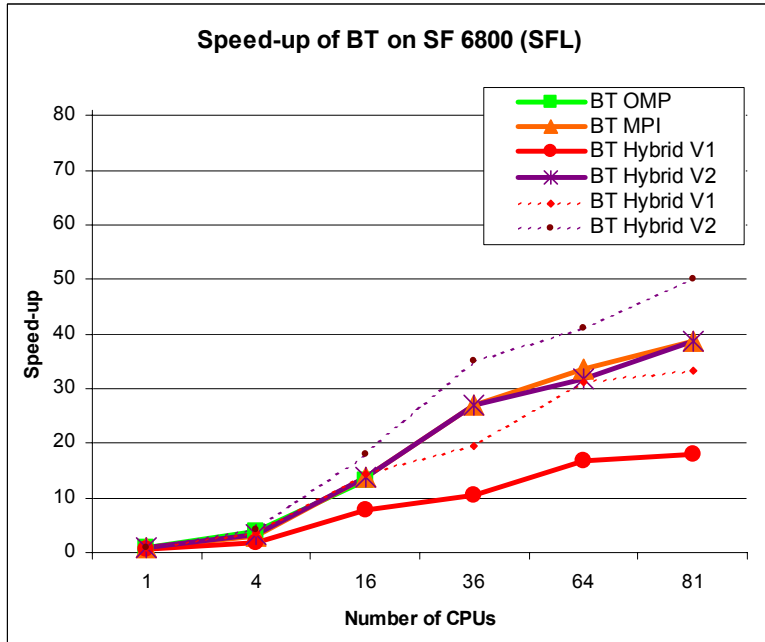
```
do ib = 1, nblock
  call receive
  !$omp parallel do
    do j=j_low,j_high
      do i=i_low,i_high
        do k=k_low,k_high
          rhs(i,j,k,ib)=
            rhs(i,j,k-1,ib)+ . . .
        enddo
      enddo
    enddo
  call send
end do
```

Speed-up for BT Class A (Fast Networks)



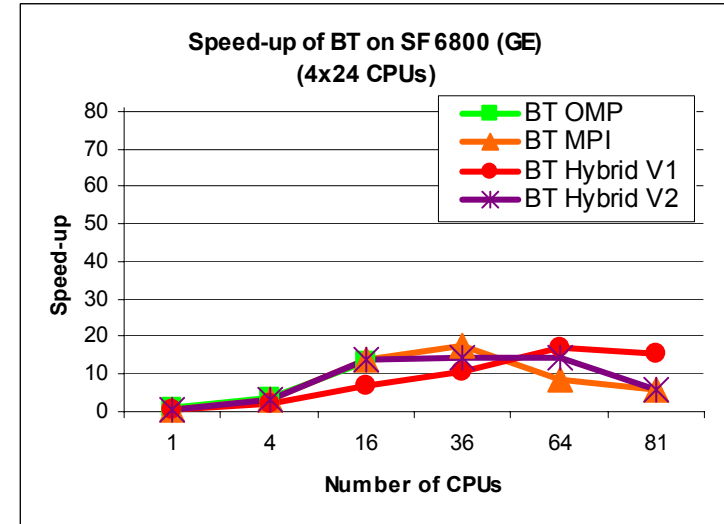
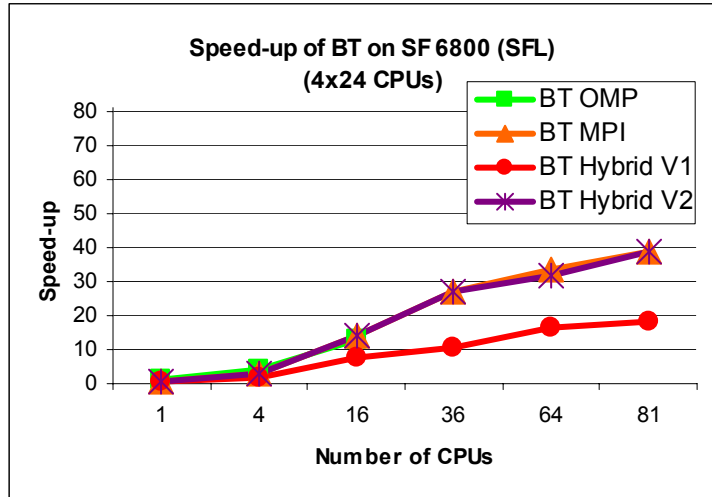
- **Problem size 64 x 64 x 64 grid points**
- **Speed-up:**
 - ▶ Measured against time of fastest implementation on 1 CPU (BT OMP)
 - ▶ For the hybrid codes the best combination of MPI processes and threads is reported
- **BT OMP: Scalability limited by**
 - ▶ 24 on SF 6800 (available number of CPUs)
 - ▶ 62 on SF 15K (due to problem size)

Speed-up for BT Class A (Fast Networks)



- **Speed-up:**
 - ▶ BT Hybrid V2:
 - Best performance with only 1 thread per MPI process:
 - No benefit employing hybrid programming paradigm
 - ▶ BT Hybrid V1:
 - Best performance for 16 MPI processes and 4 or 5 threads
 - Limitation on number of MPI processes due to implementation
 - ▶ Dashed lines indicate the speed-up compared to 1 CPU of the same implementation
 - ▶ BT MPI shows best scalability

Speed-up Comparison on Fast and Slow Network



- **Performance using Gigabit Ethernet (GE):**

- ▶ Hybrid implementations outperform pure MPI implementation
- ▶ BT Hybrid V1 shows best scalability
- ▶ BT Hybrid V1 achieves best performance employing 16 MPI processes and 4 or 5 threads respectively

Characteristics of Hybrid Codes

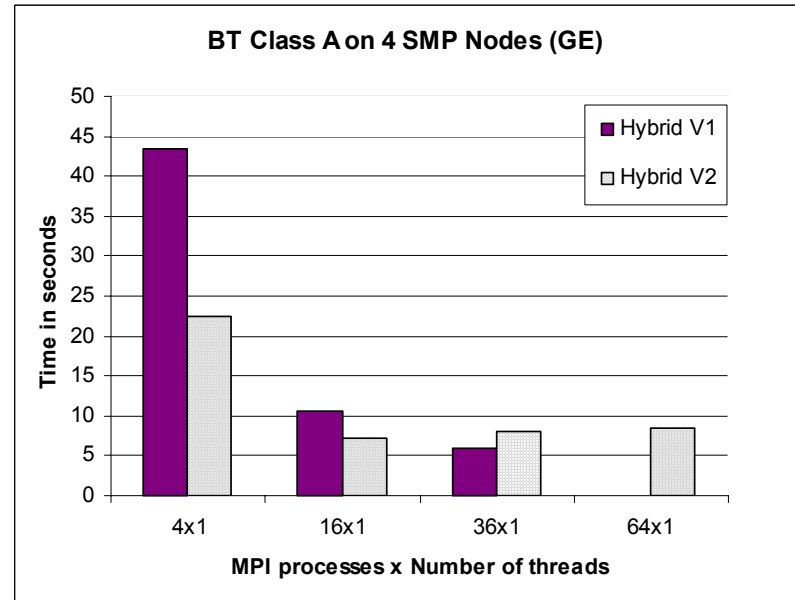
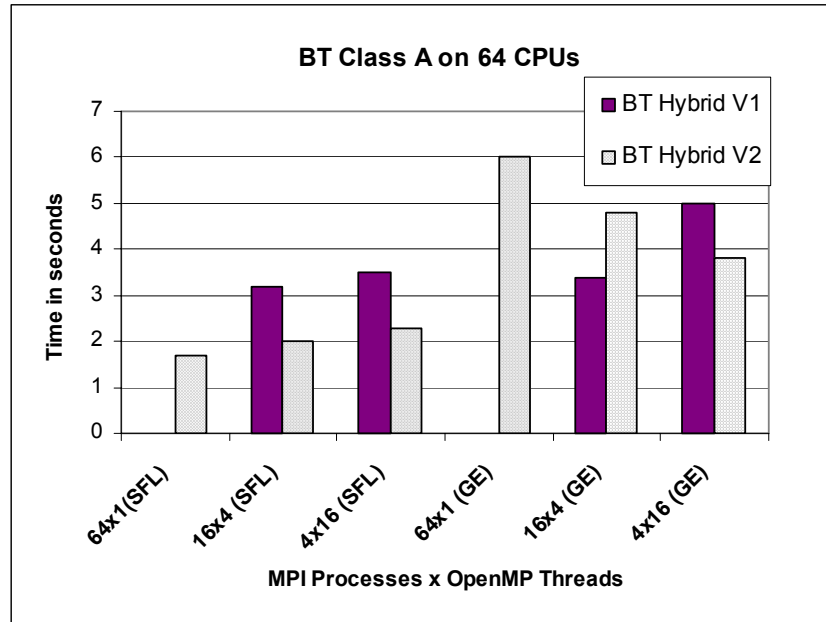
- **BT Hybrid V1:**

- ▶ Message exchange with 2 neighbor processes
- ▶ Many short messages
- ▶ Length of messages remain the same
- ▶ Increase number of threads:
 - Increase of OpenMP barrier time (threads from different MPI processes have to synchronize)
 - Increase of MPI time (MPI calls within parallel regions are serialized)

- **BT Hybrid V2:**

- ▶ Message exchange with 6 neighbor processes
- ▶ Few long messages
- ▶ Length of messages decreases with increasing number of processes
- ▶ Increase number of threads:
 - Slight increase of OpenMP barrier time
 - MPI time not affected since all communication occurs outside of parallel regions

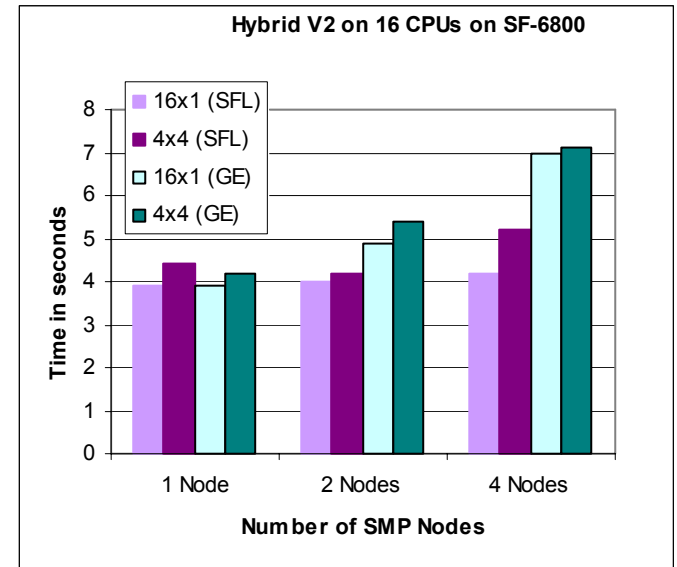
Performance of Hybrid Codes



- **BT Hybrid V1: Tight interaction between MPI and OpenMP limits number of threads that can be used efficiently**
- **BT Hybrid V2: saturates the slow network with small number of MPI processes**

Using Multiple Nodes

- **Sun Fire Link imposes little overhead when using multiple SMP nodes**



Conclusions

- **For the BT benchmark:**
 - ▶ Pure MPI implementation shows best scalability when fast network of shared memory is available
 - All CPUs within the cluster can be use
 - Coarse grained well balanced workload distribution
 - ▶ Hybrid programming paradigm advantageous for slow network
 - ▶ Tight interaction between MPI and OpenMP problematic
- **OpenMP paradigm introduces limitations to the scalability:**
 - ▶ Problem size in one dimension
 - Nested parallelism?
 - ▶ CPUs within one SMP node
 - Software DSM?
 - ▶ Mandatory barrier synchronization at the end parallel regions
 - OpenMP extensions?
 - ▶ Hybrid programming paradigm more suitable for multi-zone codes

Related Work

- **Comparison of MPI and shared memory programming (H. Shan et. all)**
- **Aspects of hybrid programming (R. Rabenseifner)**
- **Evaluation of MPI on the Sun Fire Link (S. J. Sistare et. all)**