Optimization of LESlie3D for the Cray X1 Architecture

Sam Cable and Tom Oppe, ERDC MSRC
Vicksburg, MS

- We compare execution speeds and profiles of the CFD code LESlie3D ….
  - …. On the X1, optimized code vs. original.
  - …. On the X1 vs. on the IBM POWER4.

CUG 2004

The ERDC X1: “Diamond”

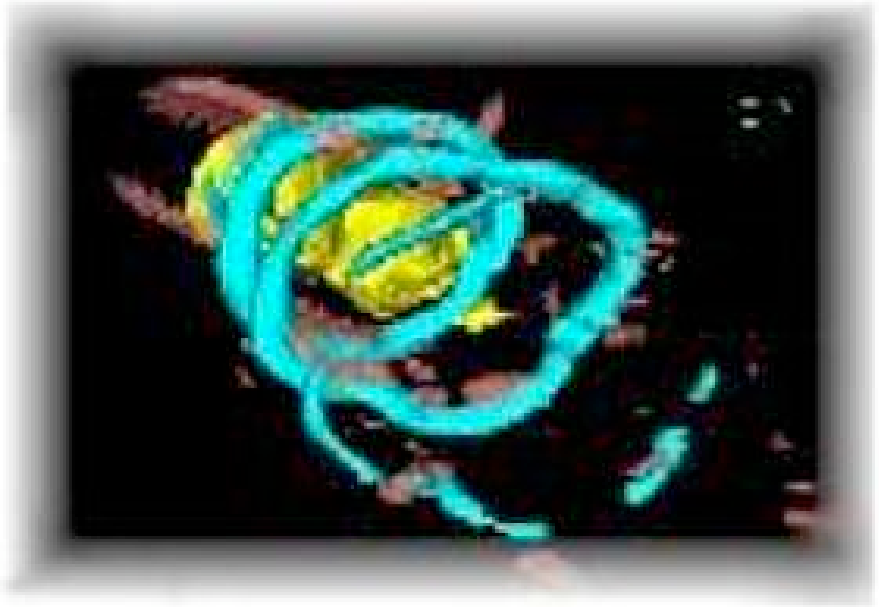
- 64 Cray X1 MSPs (256 SSPs)
- 60 MSPs available for computation (240 SSPs)
- 16 nodes of 4 MSPs each
- Liquid cooled
- 4.5 TB scratch space

CUG 2004

The Code: LESlie3D

Large Eddy Simulation Linear Eddy Model in 3D

- 3D Navier-Stokes solver for turbulent reacting flows (e.g. combustion), developed by Ga. Tech's Computational Combustion Laboratory
- Finite-volume predictor-corrector conservative scheme
- 4th order accurate in space; 2nd order in time
- Direct simulation mode and Large Eddy Simulation modes available



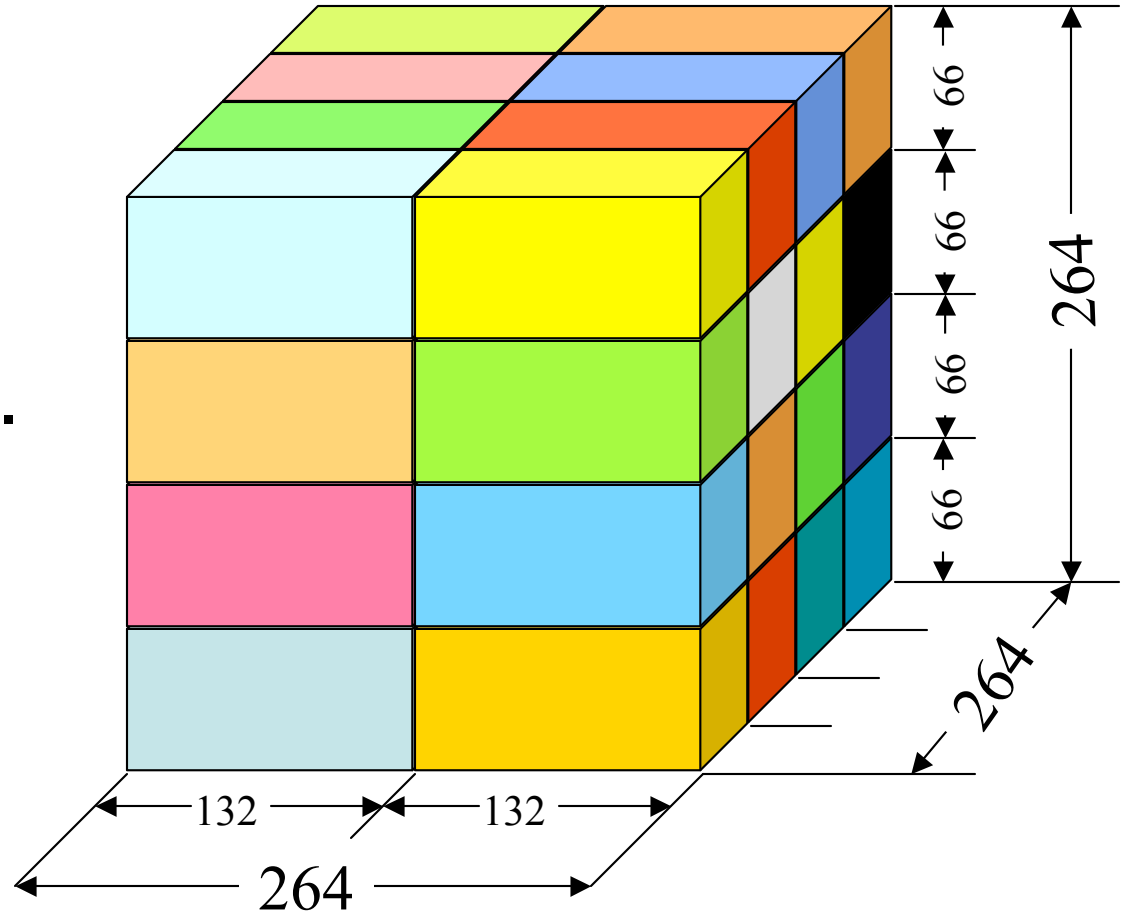
CUG 2004

LESlie3D Divides Its Structured Mesh Into Several Structured Sub-meshes.

EXAMPLE:

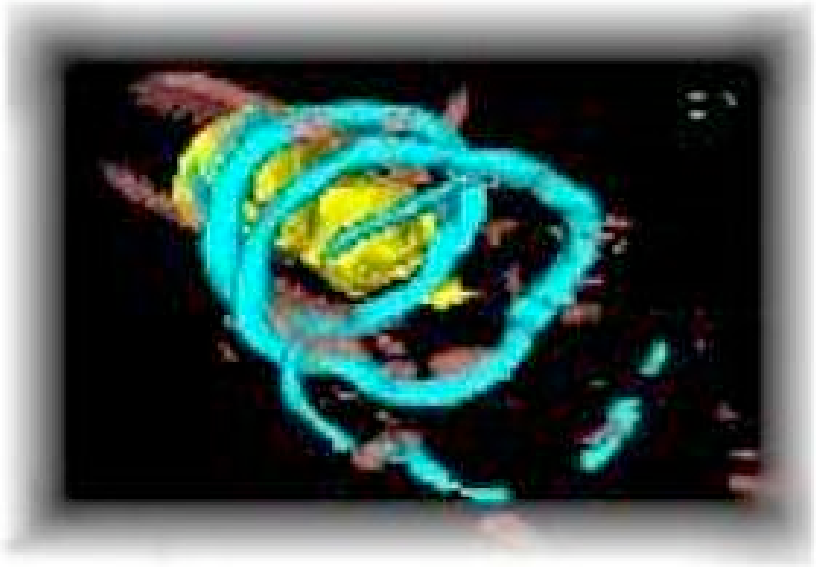
264^3 mesh

With 2x4x4 PEs.



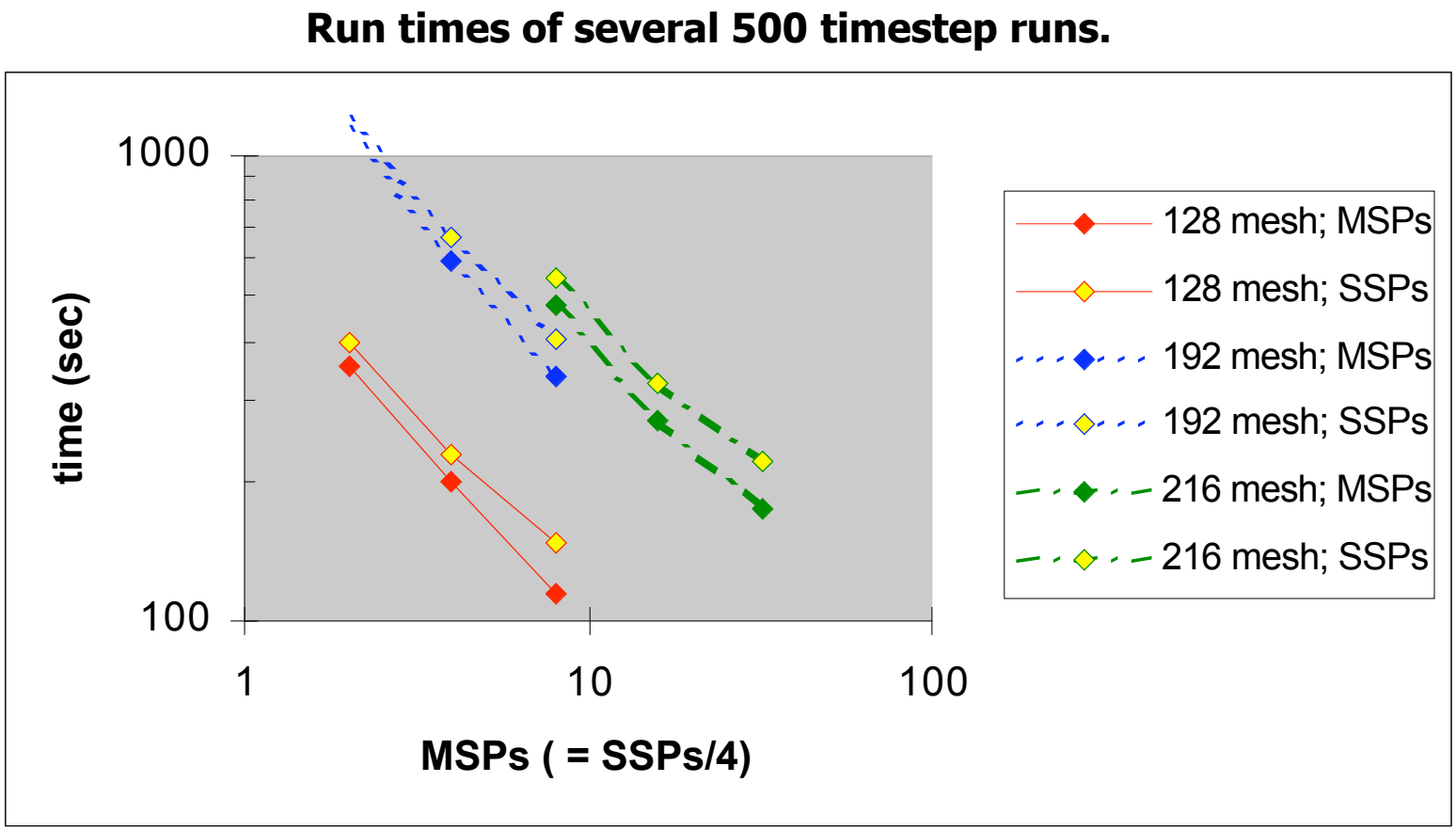
CUG 2004

LESlie3D is a vector code, which makes it a good candidate for porting to the X1.



CUG 2004

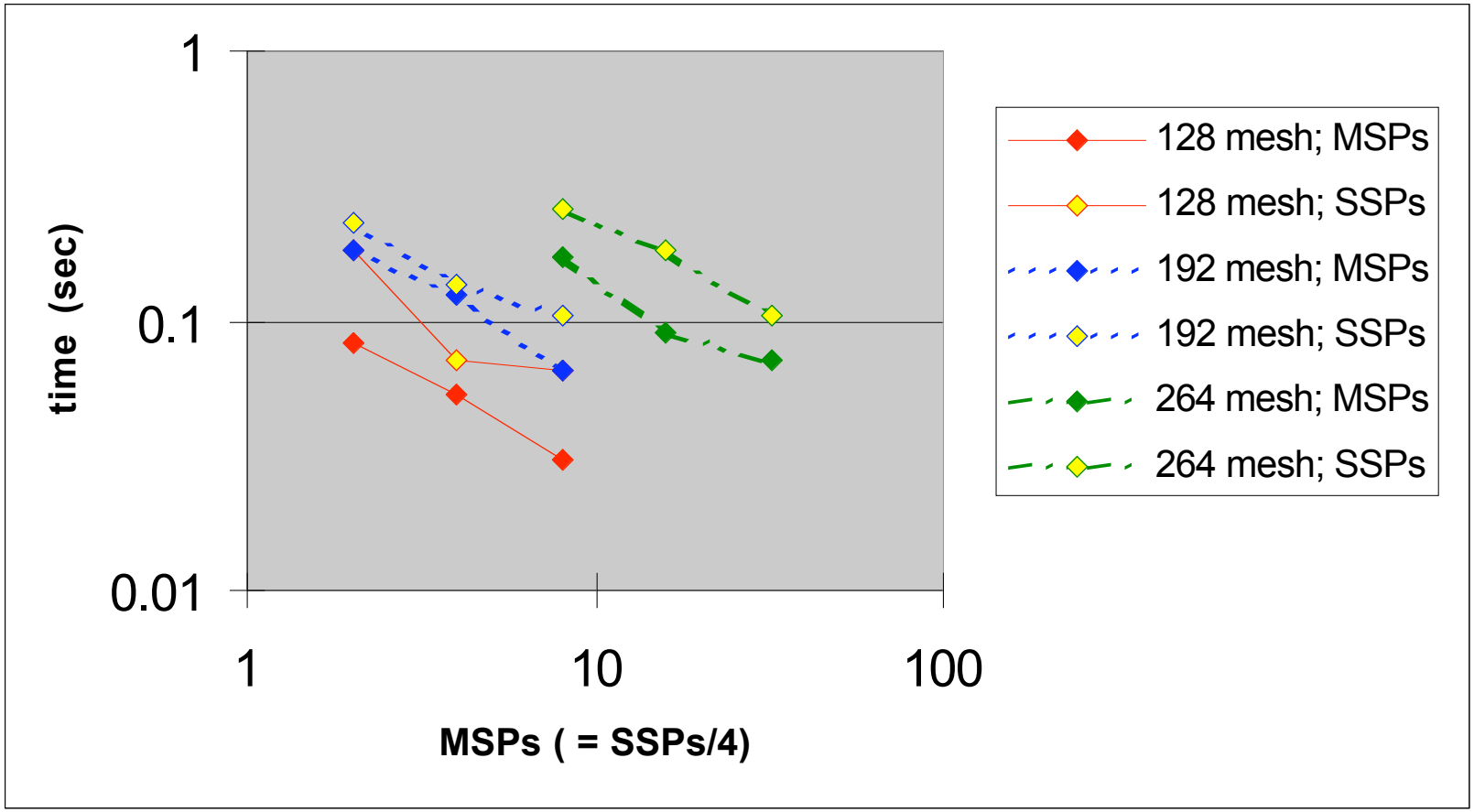
LESlie3D consistently runs faster in MSP mode than in SSP mode.



CUG 2004

MSP mode spends less time in communication.

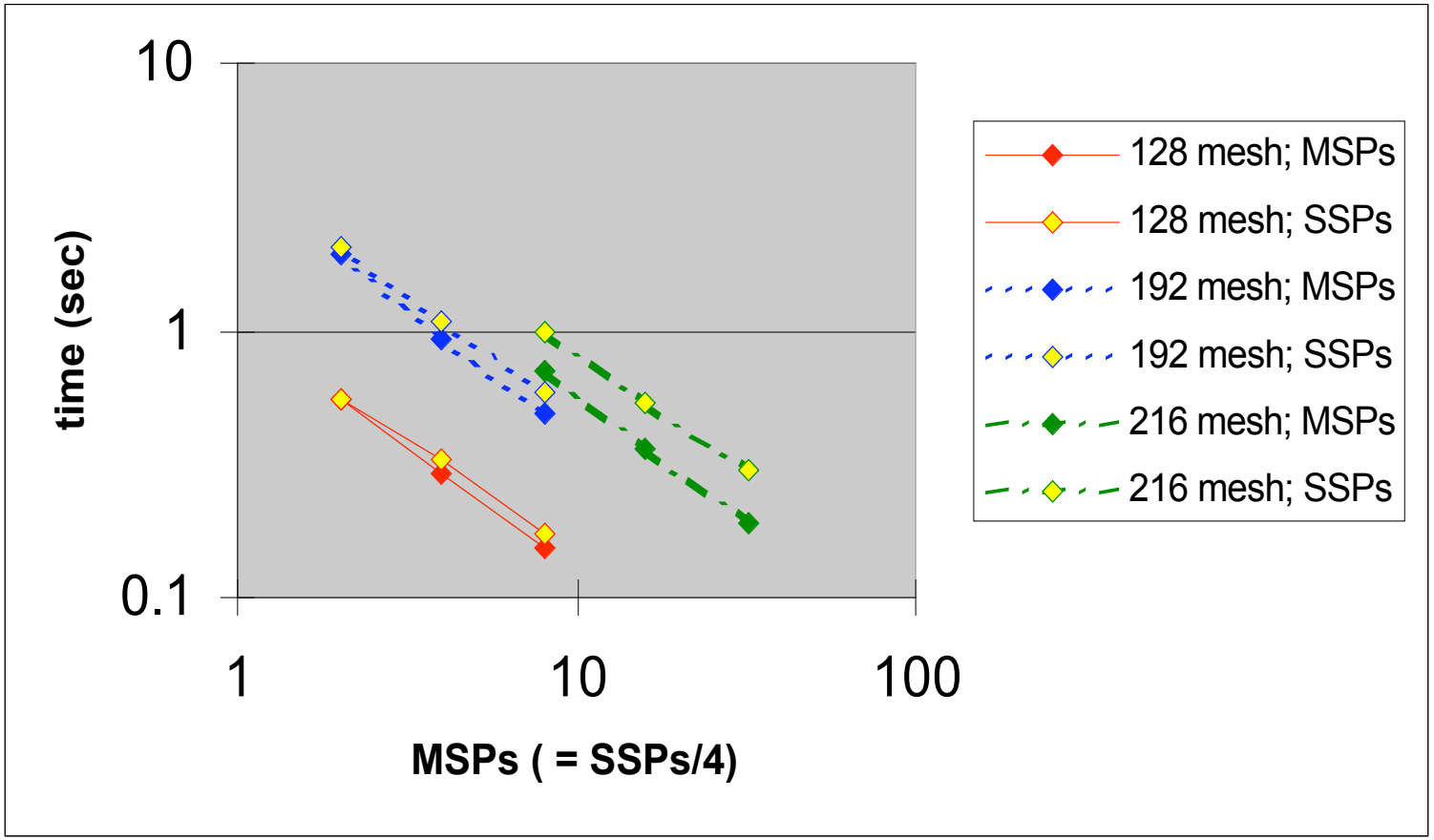
Communication time per timestep.



CUG 2004

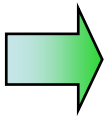
MSP runs are also faster in calculation time.

Calculation time per timestep.



Cray analysts added streaming directives to several loops to obtain multistreaming.

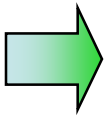
```
do k = k1,k2
  do j = j1,j2
    do i = i1,i2
           (work)
    end do
  end do
end do
```



```
!CSD$ PARALLEL DO
                PRIVATE(vars)
do k = k1,k2
  do j = j1,j2
    do i = i1,i2
           (work)
    end do
  end do
end do
!CSD END PARALLEL DO
```

Cray analysts broke up other loops to promote load balancing.

```
do nn = 1,nspeci
do k = k1,k2
  do j = j1,j2
    do i = i1,i2
         (work)
    end do
  end do
end do
end do
```



```
!CSD$ PARALLEL DO
                PRIVATE( vars )
   do kth=1,4
      k1new=k1+(k2-k1+4)/
                      4*(kth-1)
     k2new=k1new+(k2-k1+4)/
                         4 - 1
    do nn = 1,nspeci
       do k = k1new,k2new
         do j = j1,j2
           do i = i1,i2
          (work)
           end do
         end do
       end do
      (other nested loops)
     end do
  !CSD$ END PARALLEL DO
```

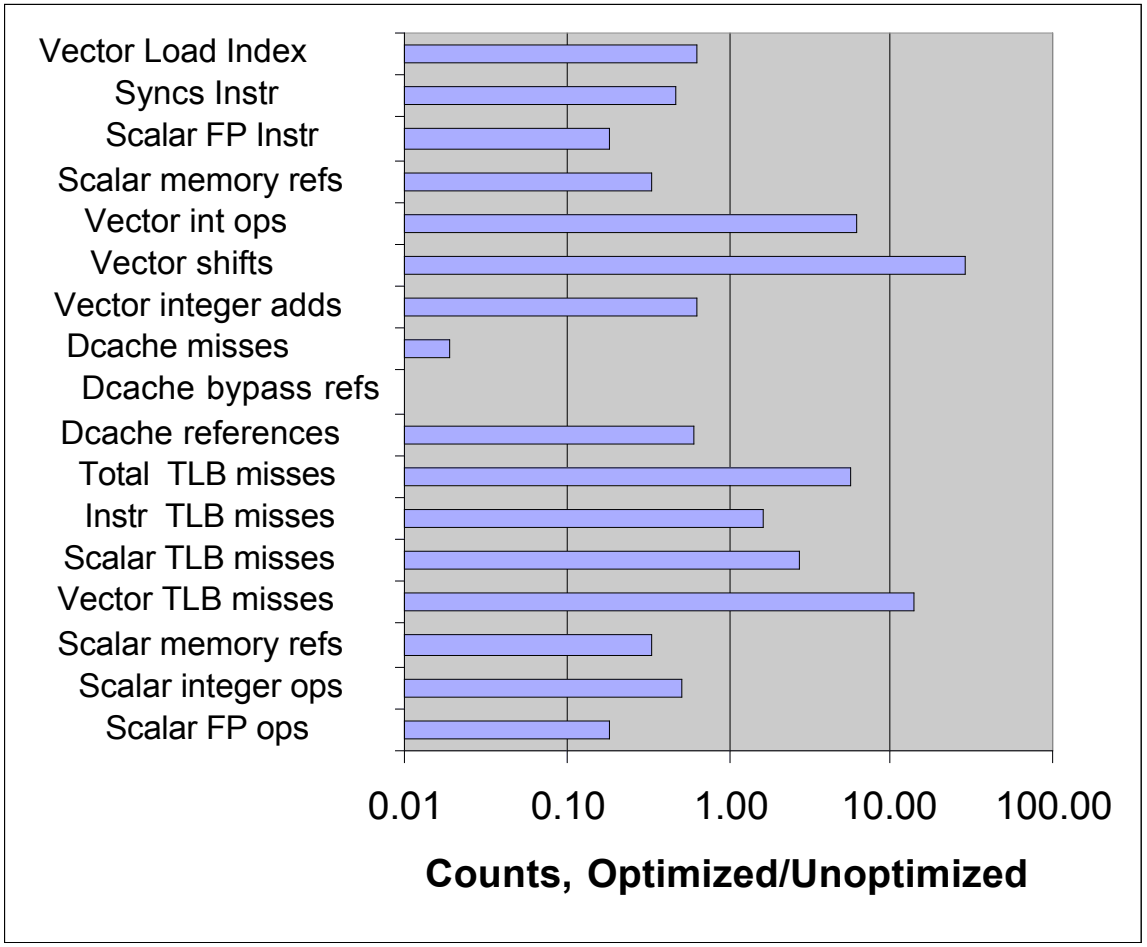
Cray CSD optimizations sped up LESlie3D by about 20 per cent.

	Run time		% speedup	% time in computation		% time in communication		% time in stats	
	Original	Optimized		Original	Optimized	Original	Optimized	Original	Optimized
192^3 mesh; 16 MSPs; 500 timesteps	3:29	2:47	20.10%	84.46%	78.95%	11.06%	15.42%	4.47%	5.63%
264^3 mesh; 32 MSPs; 4000 timesteps	0:44:35	0:35:56	19.40%	83.78%	76.26%	12.44%	18.99%	3.78%	4.75%

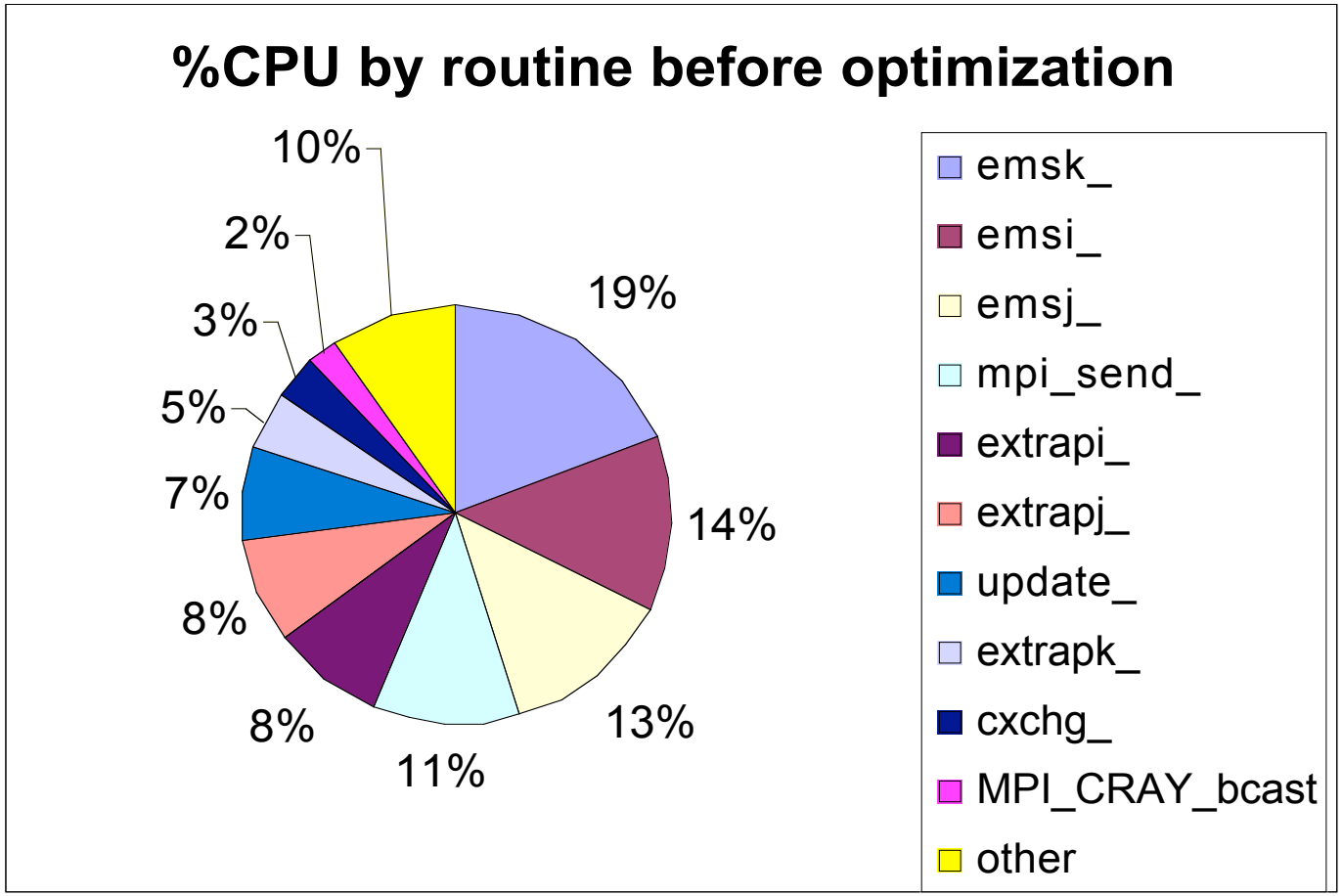
CUG 2004

Optimization generally increased vector operations and decreased scalars.

From runs on 192^3 mesh, 1x4x4 PEs

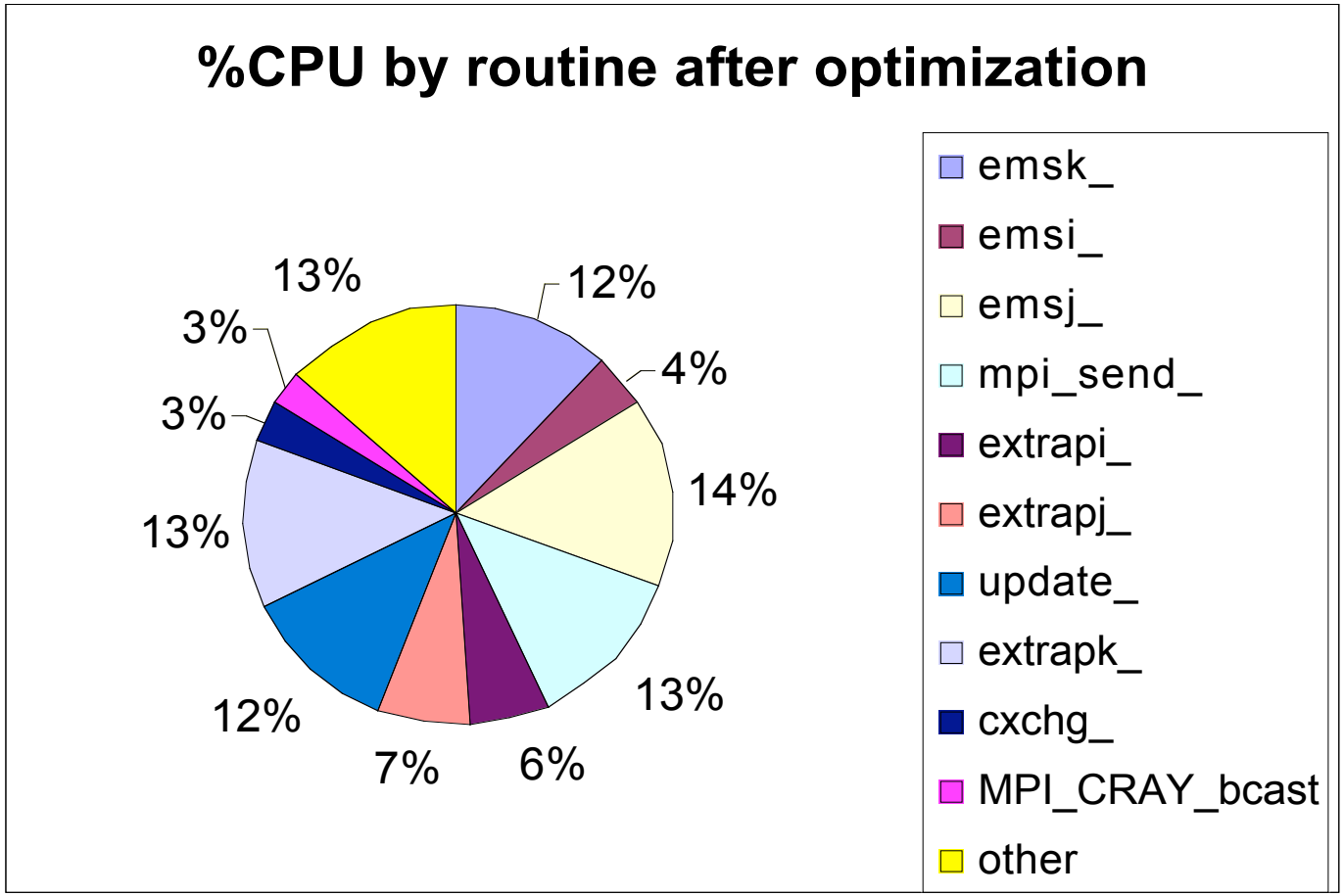


Optimization created significant changes in performance of individual routines.



CUG 2004

Optimization created significant changes in performance of individual routines.



We ran LESlie3D on the X1 and the IBM P4 and compared timings.

From 192^3 mesh; 16 PEs; 500 timesteps

	X1	X1 * 4	% of total	P4	% of total
Run time (sec)	2:47	11:08		36:34	
Time per timestep	0.334	1.336	100%	4.392	100%
Calc. time per timestep	0.264	1.056	79%	3.430	78%
Comm. time per timestep	0.052	0.208	16%	0.471	11%
Stat. time per timestep	0.019	0.076	6%	0.490	11%

We have also collected hardware performance data on the Cray X1 and the IBM P4.

- X1:
- pat_hwpc counters
  - Cycles
  - Instructions graduated
  - Vector Instructions
  - Scalar memory refs
  - Vector FP adds

- P4:
- PAPI counters
  - PM_CYC
  - PM_BR_ISSUED
  - PM_FPU_ALL
  - PM_FXU_ALL
  - PM_LD_REF_L1
  - PM_FPU_FMA

Comparison is not so straightforward….

… but we did our best.

From 192^3 mesh; 16 PEs; 500 timesteps

X1 pat_hwpc counter	counts/MSP	P4 PAPI counter	counts/PE
Cycles	2.800E+11	PM_CYC	2.831E+12
Branches & Jumps	1.384E+09	PM_BR_ISSUED	1.052E+11
Branches mispredicted	3.554E+07	PM_BR_MPED_CR+ PM_BR_MPRED_TA	2.178E+09
Total FP ops	3.524E+11	PM_FPU_ALL	1.705E+11
Vector int ops + Scalar int ops	1.117E+09	PM_FXU_FIN	3.335E+11
Dcache refs	8.715E+08	PM_LD_REF_L1	3.398E+11

Conclusion

- Optimized vs. original code
  - Optimized code ran 20% faster than original.
  - Optimizations resulted in more vector operations.
- X1 vs. P4
  - The X1 outperforms the P4 by a factor of 12 at MSP level, factor of 3 at SSP level.
  - The X1 gets better hardware performance in several areas:
    - Integer ops
    - Cache
    - Branches