LESSONS LEARNED FROM 1-YEAR EXPERIENCE WITH SX-9 AND TOWARDS NEXT-GENERATION VECTOR COMPUTING

Hiroaki Kobayashi

Cyberscience Center
Tohoku University
6-3 Aramaki-Aza-Aoba,
Sendai 980-8578, JAPAN
koba@isc.tohoku.ac.jp

Through my talk, I would like to share with you the 1-year experiences with SX-9, which is the latest vector system installed at Tohoku University in 2008. I will start with my talk to show you the HPC challenge benchmark results of our SX-9. The HPC challenge benchmark suit is designed for comprehensive benchmarking of high-end systems, and is more focusing on evaluation of sustained memory and network bandwidth, and sustained performance of some representative kernels such as FFT, which cannot clearly be evaluated by LINPACK (HPL) only. The SX-9 system achieved 19 top one scores out of 28 HPC challenge benchmark tests. We also discuss performance-tuning options for SX-9, especially those for an on-chip, software-controllable cache, which is newly introduced into the SX-9 vector processor to cover its limited off-chip memory bandwidth. We exploit the locality of vector data reference in differential equations and indirect memory accesses through list vectors in some leading scientific and engineering applications. Finally, I will introduce you our on-going research on design of the next-generation vector processor, in which multiple vector-cores sharing an on-chip cache are implemented. The preliminary performance evaluation of the multi-vector-core processor is also discussed.

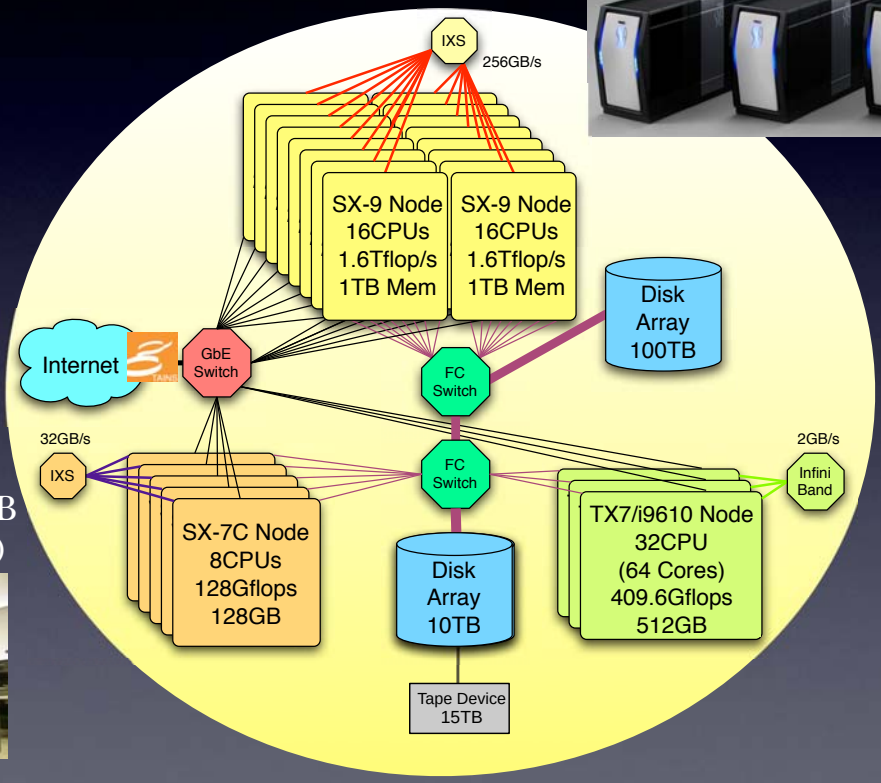
TOHOKU
UNIVERSITY

Agenda

- Lessons Learned from our 1-year experiences of SX-9
  - HPCC Benchmark Results
  - Tuning approaches to highly-efficient vector processing with caching
- Towards Next Generation Vector Computing
  - Multi Vector-Core Processor
- Summary

Supercomputers of Tohoku University

SX-9 (16-node)
26.2Tflop/s, 16TB
(Installed in 2008)



SX-7C (5-node)
640Gflop/s, 640GB
(Installed in 2006)

TX7/i9610 (3-node)
1.23Tflop/s, 1.5TB
(Installed in 2006)

Features of SX-9 of Tohoku Univ.

System
26.2TF
16TB

Node
1.6TF
1TB

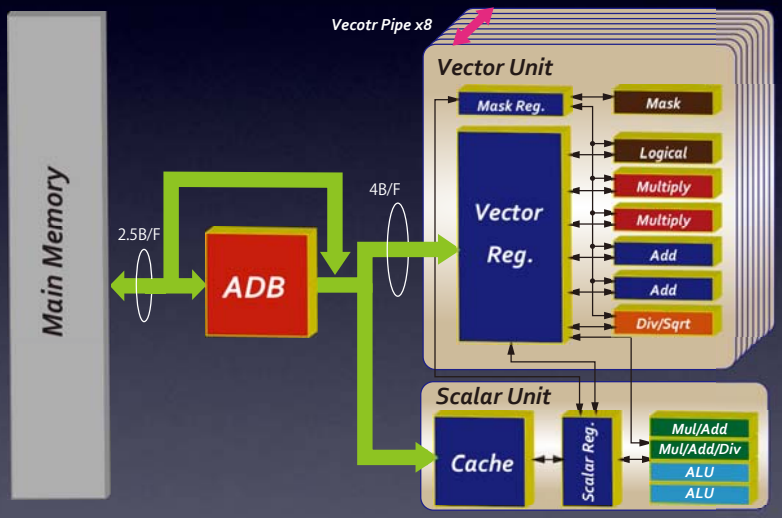
CPU
102.4GF

- High-speed inter-node custom network (128GB/s in each direction)
- Large, high-performance SMP node
- World-fastest single-chip vector processor

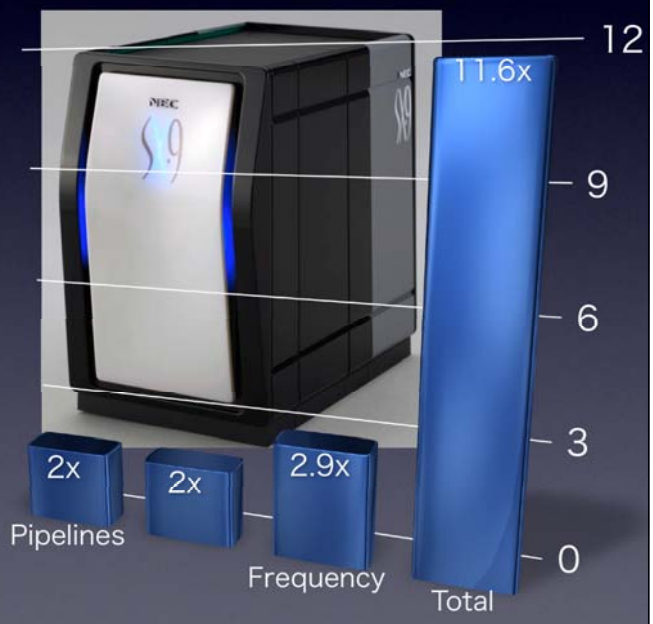
		SX-7 in 2003	SX-9 in 2008	improvement
Per CPU	Freq.	1.1GHz	3.2GHz	2.9x
	Vec. Perf.	8.83Gflop/s	102.4Gflop/s	11.6x
	Mem. BW	35.32GB/s	256GB/s	7.3x
Per SMP Node	Vec. Perf.	282Gflop/s	1.6Tflop/s	5.8x
	Mem. Cap.	256GB	1TB	4x
	Mem. BW	1.13TB/s	4TB/s	3.5x
	Mem. Banks	16K	32K	2.0x
	IXS BW	32G/s*	256GB/s	8.0x
System	Total Perf.	2.1Tflop/s	26.2Tflop/s	12.5x
	Total Mem.	2TB	16TB	8x

*SX-7C(8) in 2006

SX-9 Processor Architecture



Single Processor Performance
Improvement over 5 years (vs. SX-7)



102.4=4ops. x 8units x 3.2GHz

Hiroaki Kobayashi, Tohoku University

Performance Evaluation of SX-9 by using HPC Challenge Benchmark

HPCC Benchmark: Comprehensive Benchmarking for High-End Systems

- **Measure performance of various memory access patterns**
  - From low locality to high locality
  - More focus on Memory bandwidth and network bandwidth
    - Important factors for high SSP in execution of practical applications
- **Project leader: Jack Dongarra**
- **Sponsored by**
  - **DARPA**
    - **HPCS** program
  - **DOE**
  - **NSF**

The HPCC benchmark consists of

-for peak performance in flop/s
1. **HPL: High Performance LINPACK**
2. **DGEMM: matrix-matrix multiply**

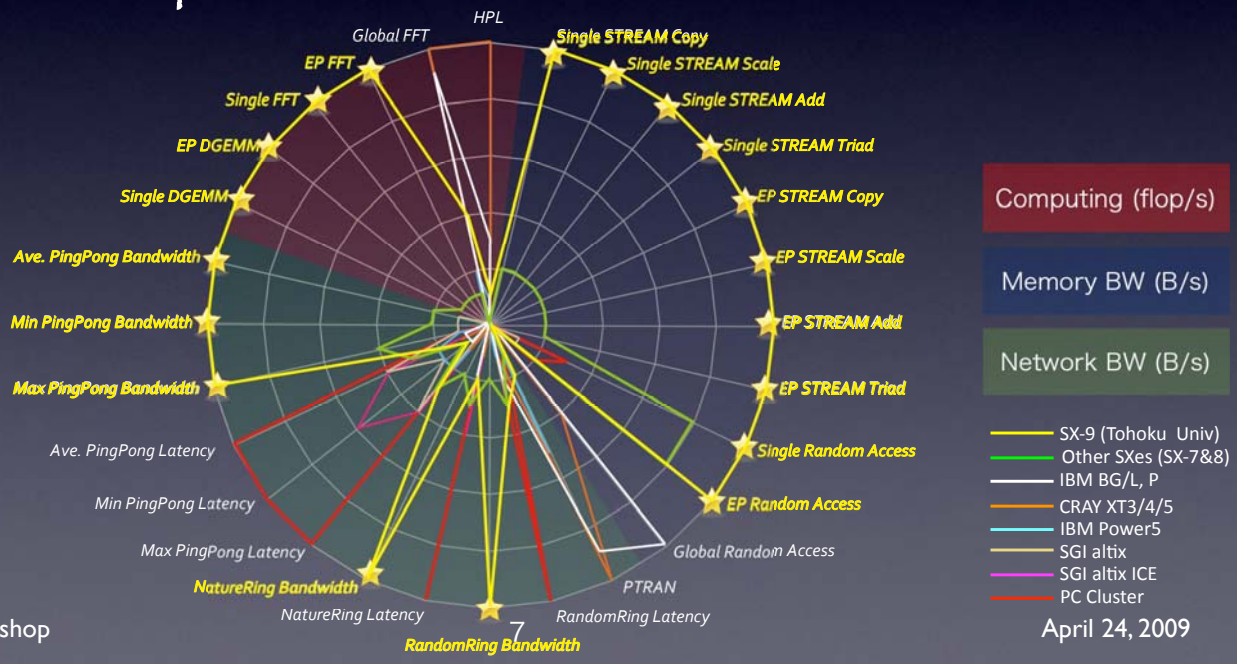
-for memory BW
3. **STREAM: simple linear algebra vector kernels (in GB/s)**
4. **RandomAccess: Irregular Memory Updates (GUP/s)**

-for communication capacity of NW
5. **PTRANS: parallel matrix transpose**
6. **Communication BW and latency: in a number of simultaneous communication patterns**

- for sustained performance on application kernel
7. **FFTE: 1D DFT performance in flop/s**

HPC Challenge Results

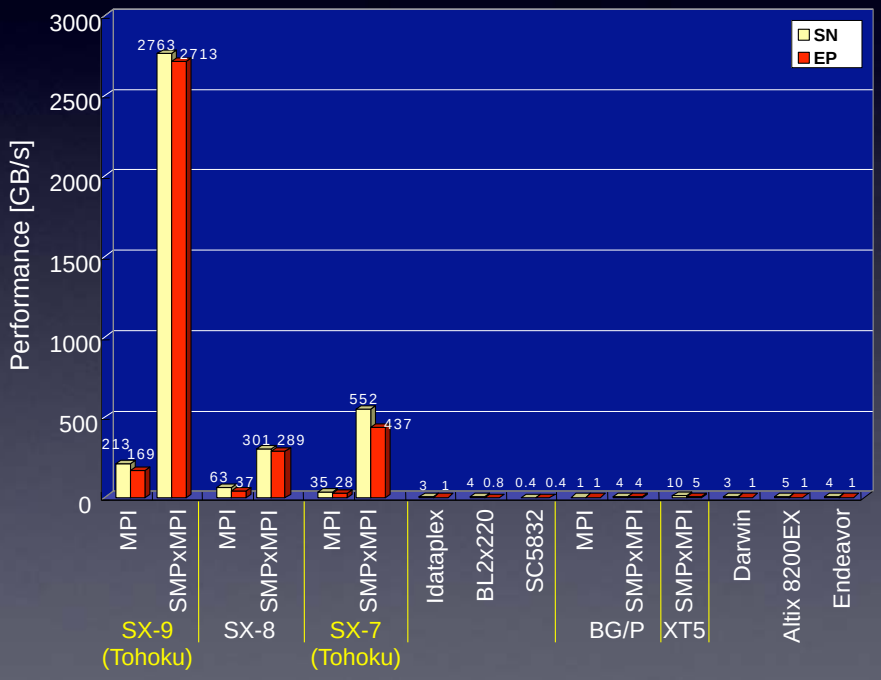
19 Top One Scores in 28 Tests (as of 2008.11.18)



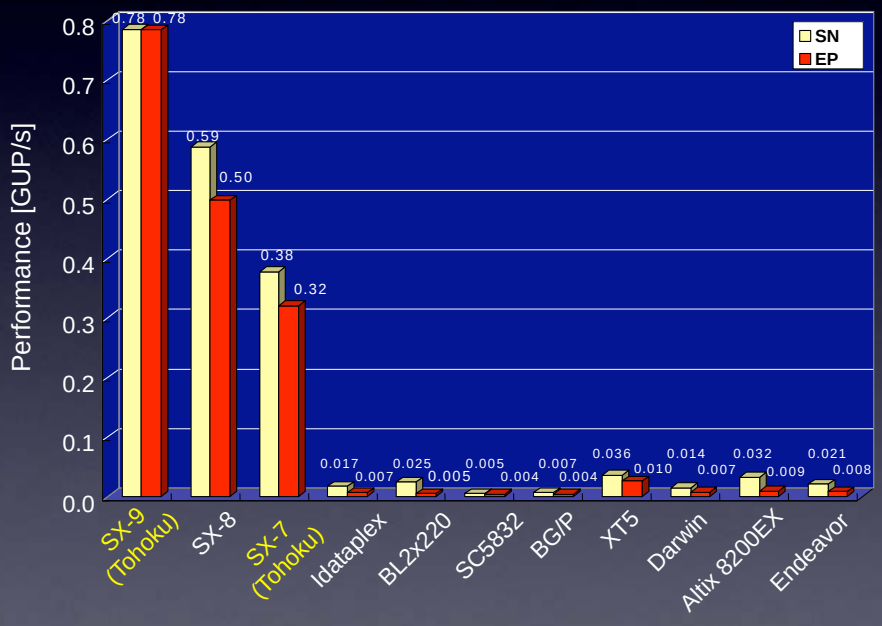
System Specifications

System name	manufacture	Processor Type	Freq.	# of Cores	# of MPI Proc	# of Threads	Peak [TF]	Interconnect Type	Network BW
SX-9	NEC	SX-9	3.2GHz	256	256	1	26.2	IXS	128GB/s
SX-9(SMP)	NEC	SX-9	3.2GHz	32	2	16	3.2	IXS	128GB/s
SX-8	NEC	SX-8	2GHz	40	40	1	0.64	IXS	16GB/s
SX-8(SMP)	NEC	SX-8	2GHz	40	5	8	0.64	IXS	16GB/s
SX-7	NEC	SX-7	0.552GHz	32	32	1	0.28256	non	non
SX-7(SMP)	NEC	SX-7	0.552GHz	32	2	16	0.28256	non	non
Idataplex	IBM-Serviware	Xeon X5472	3.0GHz	1088	1,088	1	13.5	Infiniband	2GB/s
BL2x220	HP	Xeon E5450	3.0GHz	256	256	1	3.072	Infiniband	2GB/s
SC5832	SiCortex	SiCortex Ice9	0.7GHz	5760	5,760	1	8.064	Custom	6GB/s
Blue Gene/P	IBM	PowePC450	0.85GHz	131,072	131,072	1	557	Torus	425MB/s
Blue Gene/P (SMP)	IBM	PowePC450	0.85GHz	131,072	32,768	4	557	Torus	425MB/s
XT5	CRAY	AMD Opteron	2.3GHz	149,058	74,529	2	1,381.62	Seastar	9.6GB/s
Darwin	ClusterVision/ Dell/QLogic	Xeon 5160	3GHz	256	256	1	3.072	Infiniband	2GB/s
Altix 8200EX	SGI	Xeon X5472	3GHz	1,024	1,024	1	12.288	Infiniband	2GB/s
Intel Endeavor cluster	Intel	Xeon 5160	3GHz	1,024	1,024	1	11.4688	Infiniband	2GB/s

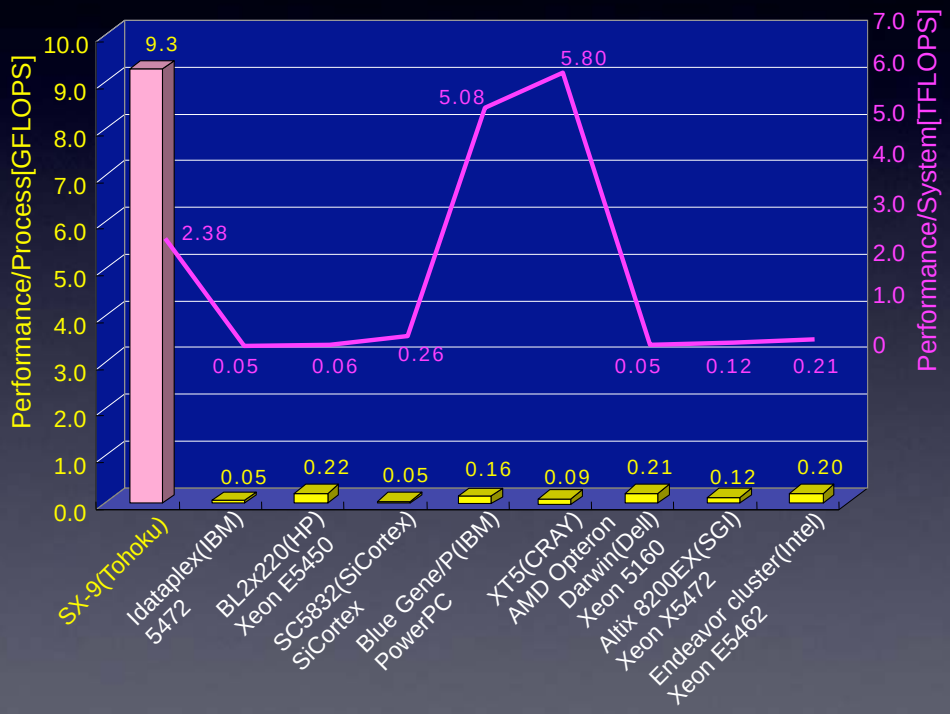
STREAM (Averaged)



Random Access (SN/EP)



G-FFT



G-FFT Results:Highly Efficient Computing on SX-9

(As of 2008.11.18)

RANK	System	Institution	Peak Perf. (Tflop/s)	CPUs (Cores)	G-FFT Results (Tflop/s)	Efficiency
1	Cray XT5	Oak Ridge National Lab.	1381.6	37544 (150176)	5.8	0.4%
2	BlueGene/P	Argonne National Lab.	557	32768 (131072)	5.1	0.9%
3	Red/Storm/XT3	Sandia National Lab.	124.4	12960 (25920)	2.9	2.3%
4	SX-9	Tohoku Univ.	26.2	256 (256)	2.3	9.1%

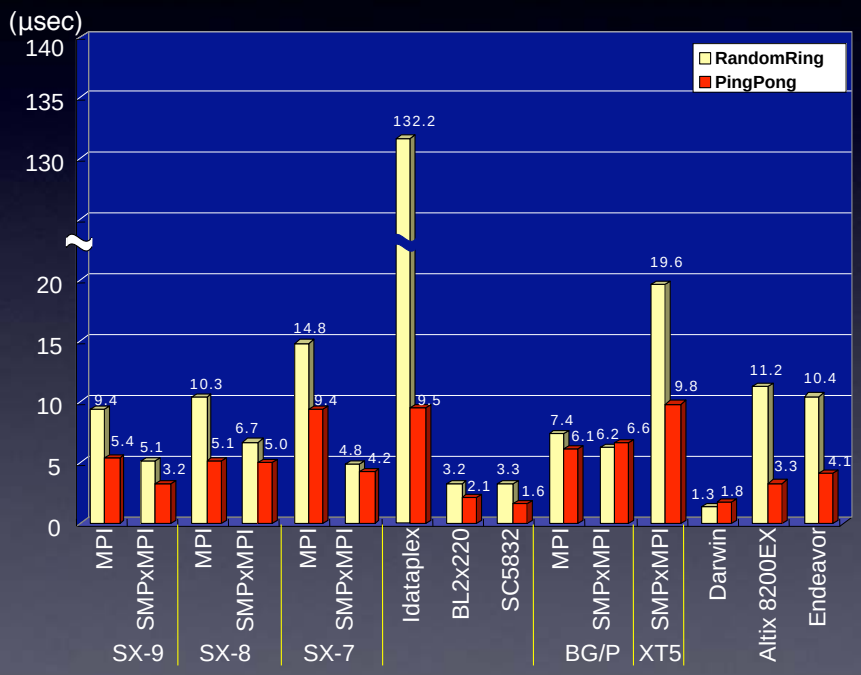
52.7x

587x

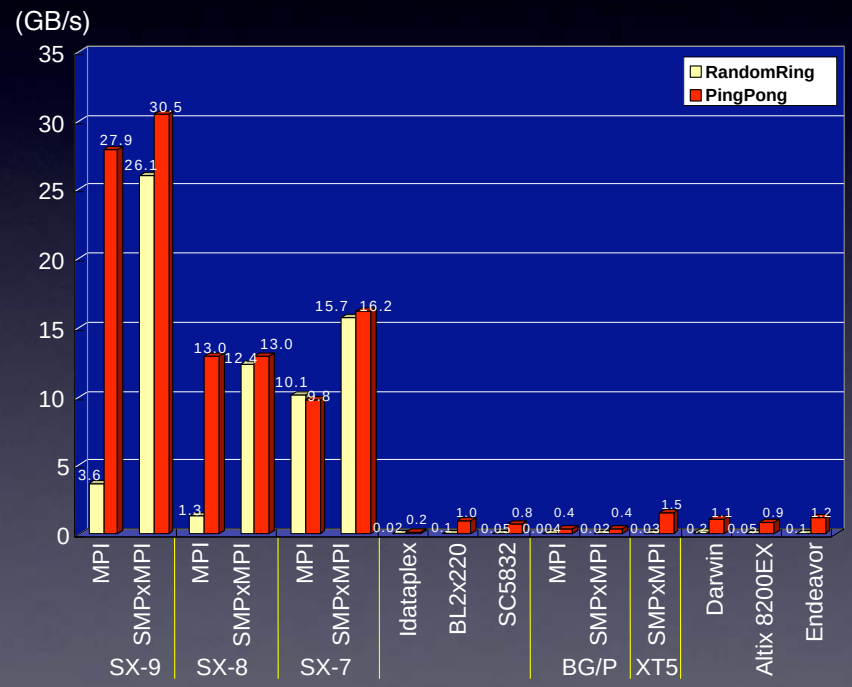
Only 2.5x!

23x

Latency

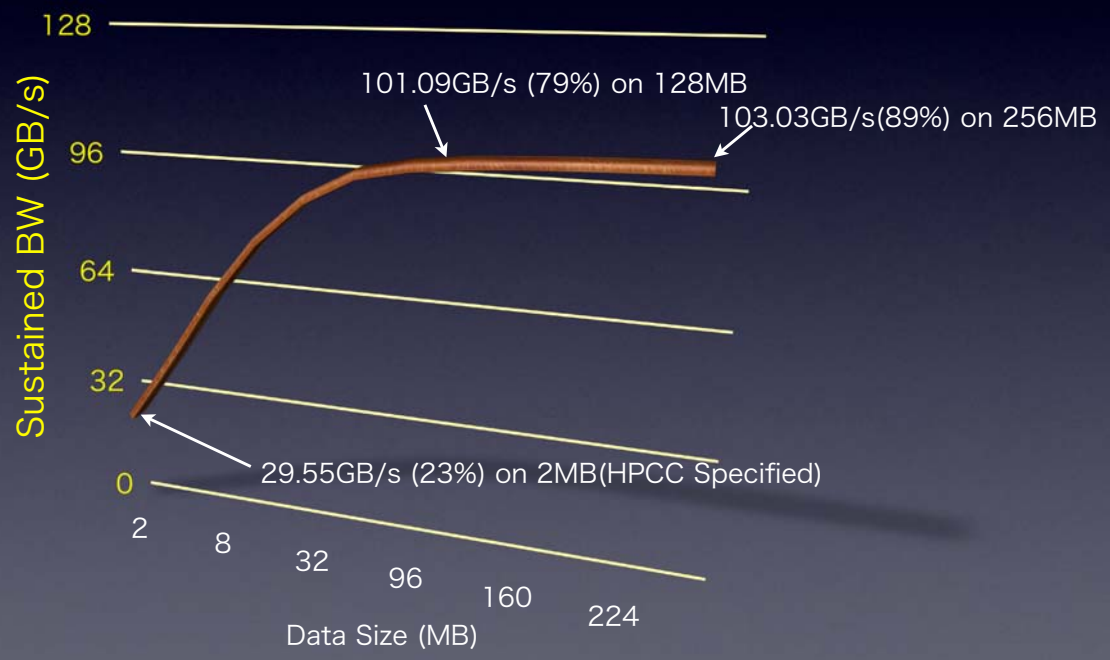


Bandwidth

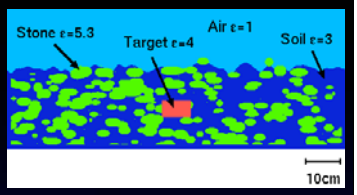


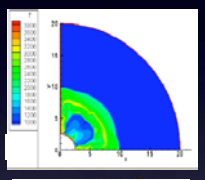
PingPong Performance of SX-9 IXS

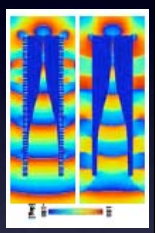
- Peak: 128 GB/s in each direction



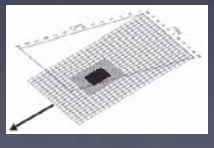
Discussion on Tuning Techniques for SX-9





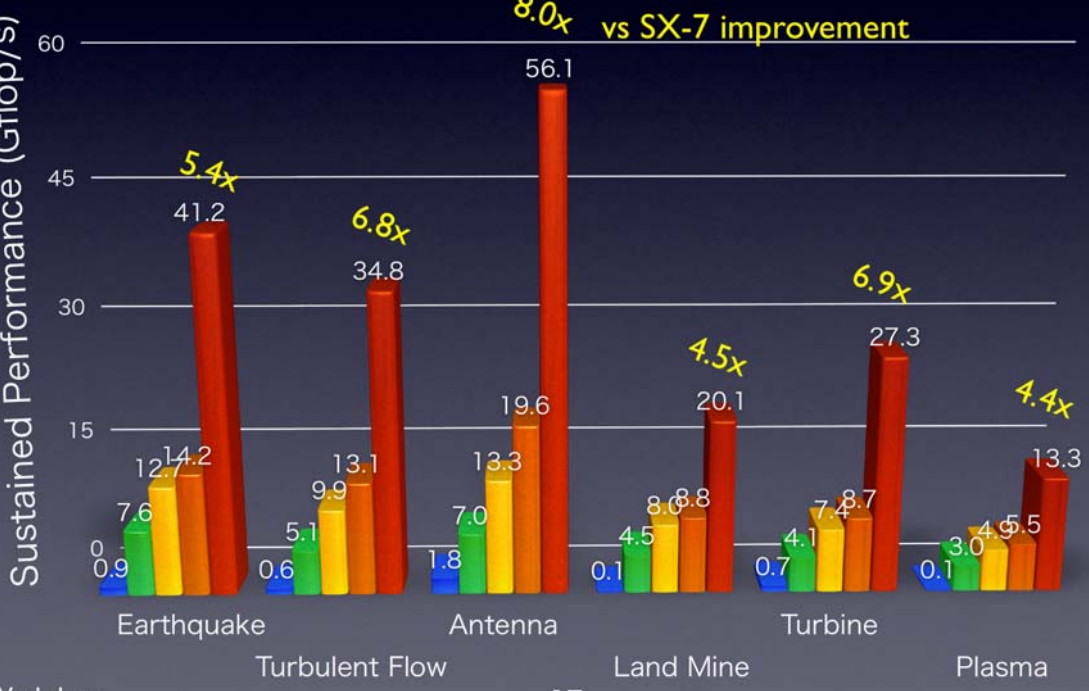


- Points for tuning
  - effect of 2B/F from 4B/F
    - be aware of high-vector processing rate, relatively lower memory bandwidth
    - increase computations and reduce memory operations as many as possible
  - effect of 256KB ADB
    - figure out temporal locality of vector data reference
- Applications examined
  - Earthquake
    - Simulation of seismic slow slip model
  - Turbulent flow
    - Direct numerical simulation of turbulent channel flow
  - Antenna
    - FDTD simulation of lens antenna using Fourier transform
  - Land Mine
    - FDTD simulation of array antenna ground penetrating radar for land mine detection
  - Turbine
    - Direct numerical simulation of unsteady flow through turbine channels for hydroelectric generators
  - Plasma
    - Simulation of upper hybrid wave in plasma using Lax-Wendroff method



SX-9 Single CPU Performance

Itanium II	SX-7	SX-8	SX-8R	SX-9
6.4 Gflop/s	8.8 Gflop/s	16 Gflop/s	35.2 Gflop/s	102.4 Gflop/s



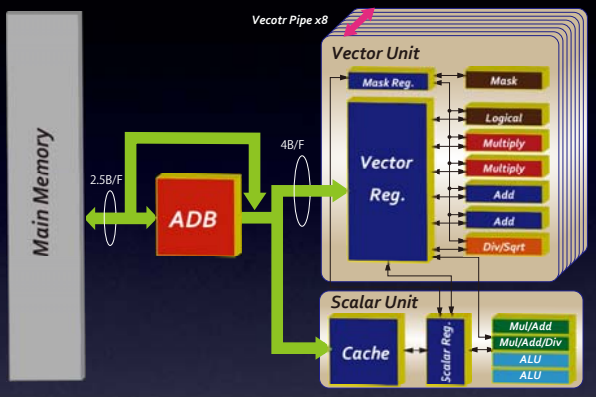
Tuning Options for Efficient Vector Processing with On-Chip Caching on SX-9

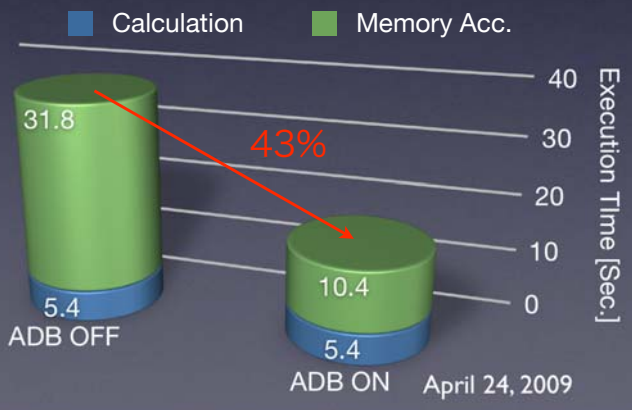
- Selective Caching
  - increasing opportunities of cache hits of data with higher temporal locality
- Cache Blocking
  - increasing cache hit rates to avoid capacity misses
  - decreasing vector length
- Loop Unrolling/Loop Fusion
  - increasing arithmetic density/vector length in loop body
  - decreasing the branch overhead
  - increasing the temporal locality of data by removing duplicated vector loads across nested loops
    - ✓ now more sensitive to SX-9 performance due to its limited memory BW
  - increasing the possibility of register spill and/or eviction from the cache if their capacities are not enough, because the data should be available on the chip for a long time
    - ★ this also give a pressure to the memory system, especially 2.5B/F of SX-9

Selective Caching & Blocking (Case 1)

Plasma with list vector accesses

```
!cdir on_adb(dvecw),nodep
      do ii=jj,min(jj+lvec-1,iplast)
        kk = ii - jj + 1
        aa     = xp1(ii)/delx
        ic     = int( aa+half )
        raa1 = ic
        dd1    = fact1*(aa+half-raa1)
        dd2    = fact1*(raa1-aa+half)
        dvecw(ic   ,kk,1)=dvecw(ic   ,kk,1)+dd1*vp1(1,ii)
        dvecw(ic-1,kk,1)=dvecw(ic-1,kk,1) +dd2*vp1(1,ii)
        dvecw(ic   ,kk,2)=dvecw(ic   ,kk,2)+dd1*vp1(2,ii)
        dvecw(ic-1,kk,2)=dvecw(ic-1,kk,2) +dd2*vp1(2,ii)
        dvecw(ic   ,kk,3)=dvecw(ic   ,kk,3)+dd1*vp1(3,ii)
        dvecw(ic-1,kk,3)=dvecw(ic-1,kk,3) +dd2*vp1(3,ii)
        dvecw(ic   ,kk,4)=dvecw(ic   ,kk,4)+dd1
        dvecw(ic-1,kk,4)=dvecw(ic-1,kk,4) +dd2
      enddo
```





Selective Caching & Blocking (Case 2)

- Multiply-add kernel for complex array

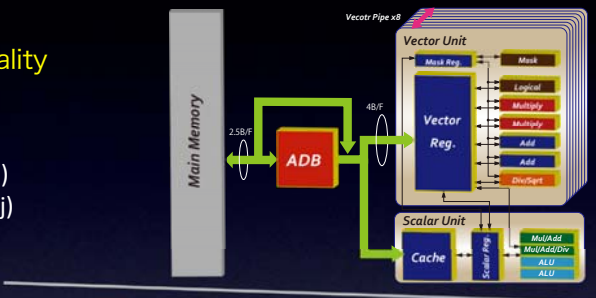
```
do k=1,l
  do j=1,m
    do i=1,n
      sum(i,1,k) = sum(i,1,k) + dble(a(j,k))*b(i,1,j) - aimag(a(j,k))*b(i,2,j)
      sum(i,2,k) = sum(i,2,k) + dble(a(j,k))*b(i,2,j) + aimag(a(j,k))*b(i,1,j)
    end do
  end do
end do
```

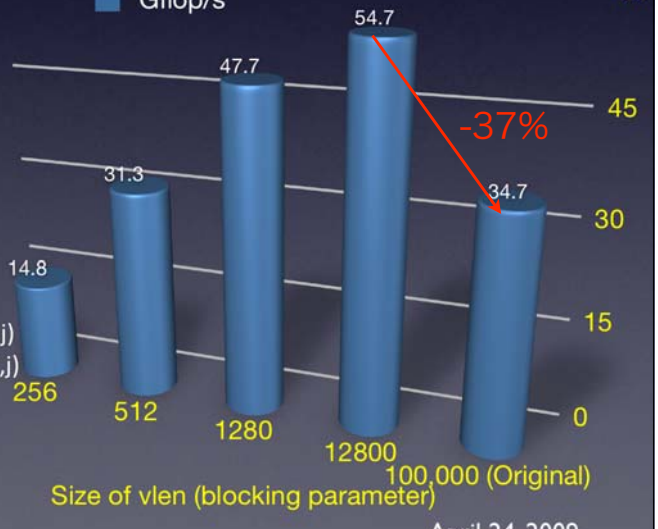
sum(i,1,k) & sum(i,2,k) have temporal locality
but they need 1.6MB cache space!

Selective caching with blocking

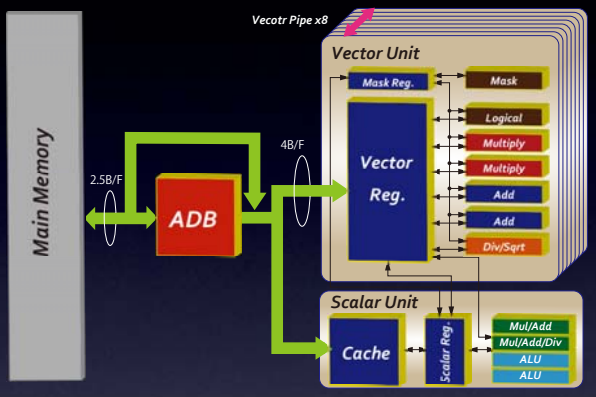
vlen : cache blocking parameter (if vlen= or < 12,800, sum(i,1,k) & sum(i,2,k) are cacheable on the 256KB ADB)

```
do k=1,l
  do i2=1,n,vlen
    do j=1,m
!cdir on_adb(sum)
      do i=i2,min(n,i2+vlen)
        sum(i,1,k) = sum(i,1,k) + dble(a(j,k))*b(i,1,j) - aimag(a(j,k))*b(i,2,j)
        sum(i,2,k) = sum(i,2,k) + dble(a(j,k))*b(i,2,j) + aimag(a(j,k))*b(i,1,j)
      end do
    end do
  end do
end do
```



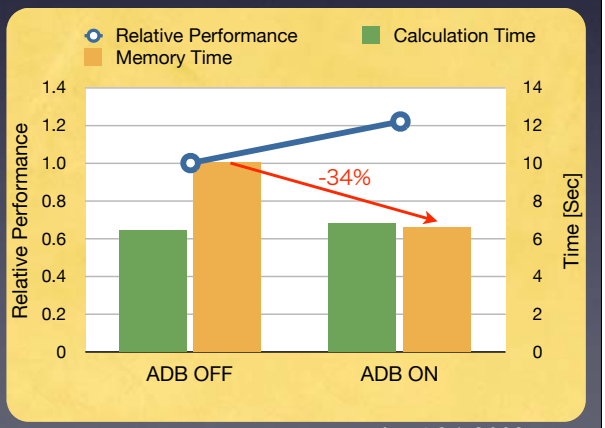


Selective Caching for Difference Equation Code (Case 3)



Land Mine (FDTD)

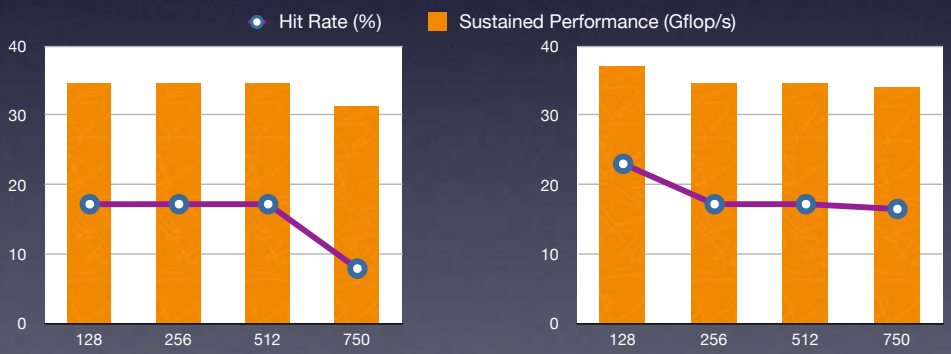
```
01    DO 10 k=0,Nz
02       DO 10 i=0,Nx
03  !cdir ON_ADB(H_x)
04  !cdir ON_ADB(H_y)
05  !cdir ON_ADB(H_z)
06          DO 10 j=0,Ny
07            E_x(i,j,k) = C_x_a(i,j,k)*E_x(i,j,k)
08  &               + C_x_b(i,j,k) * (( H_z(i,j,k) -H_z(i,j-1,k))/dy
09  &               -(H_y(i,j,k) -H_y(i,j,k-1))/dz -E_x_Current(i,j,k))
10            E_y(i,j,k) = C_y_a(i,j,k)*E_y(i,j,k)
11  &               + C_y_b(i,j,k) * ((H_x(i,j,k) -H_x(i,j,k-1))/dz
12  &               -(H_z(i,j,k) -H_z(i-1,j,k))/dx -E_y_Current(i,j,k))
13            E_z(i,j,k) = C_z_a(i,j,k)*E_z(i,j,k)
14  &               + C_z_b(i,j,k) * ((H_y(i,j,k)-H_y(i-1,j,k) )/dx
15  &               -(H_x(i,j,k)-H_x(i,j-1,k))/dy -E_z_Current(i,j,k))
16 10 CONTINUE
```



Selective Caching and Blocking: Tradeoff between Vector Length and Cache Hit Rate

Land Mine (FDTD)

```
01    DO 10 k=0,Nz
02       DO 10 i=0,Nx
03  !cdir ON_ADB(H_x)
04  !cdir ON_ADB(H_y)
05  !cdir ON_ADB(H_z)
06          DO 10 j=0,Ny
07            E_x(i,j,k) = C_x_a(i,j,k)*E_x(i,j,k) + C_x_b(i,j,k) * (( H_z(i,j,k) -H_z(i,j-1,k))/dy -(H_y(i,j,k) -H_y(i,j,k-1))/dz -E_x_Current(i,j,k))
08            E_y(i,j,k) = C_y_a(i,j,k)*E_y(i,j,k) + C_y_b(i,j,k) * ((H_x(i,j,k) -H_x(i,j,k-1))/dz -(H_z(i,j,k) -H_z(i-1,j,k))/dx -E_y_Current(i,j,k))
09            E_z(i,j,k) = C_z_a(i,j,k)*E_z(i,j,k) + C_z_b(i,j,k) * ((H_y(i,j,k)-H_y(i-1,j,k) )/dx -(H_x(i,j,k)-H_x(i,j-1,k))/dy -E_z_Current(i,j,k))
10 10 CONTINUE
```



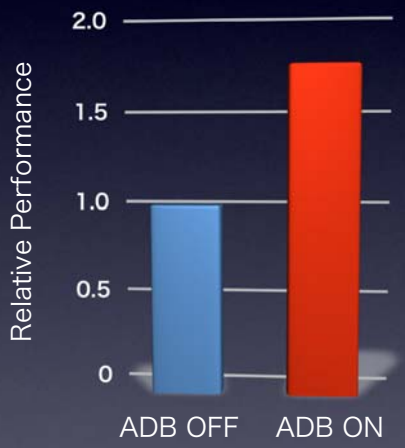
Effects of Loop Unrolling on ADB (Case 1)

Earthquake

```
do i=1,ncells
  do j=1,ncells
    wf_dip(i)=wf_dip(i)+gd_dip(j,i)*wary(j)
  end do
end do
```

Unrolling of outer loop

```
do i=1,ncells, k
  do j=1,ncells
    wf_dip(i)=wf_dip(i)+gd_dip(j,i)*wary(j)
    wf_dip(i+1)=wf_dip(i+1)+gd_dip(j,i+1)*wary(j)
       ...
    wf_dip(i+k-1)=wf_dip(i+k-1)+gd_dip(j,i+k-1)*wary(j)
  end do
end do
```



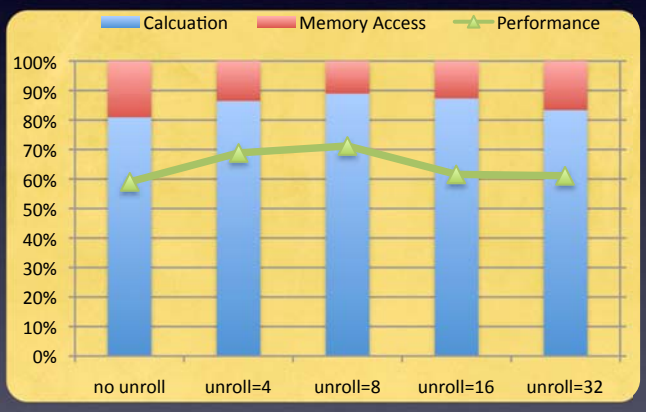
Effects of Loop Unrolling on ADB (Case 2)

Earthquake

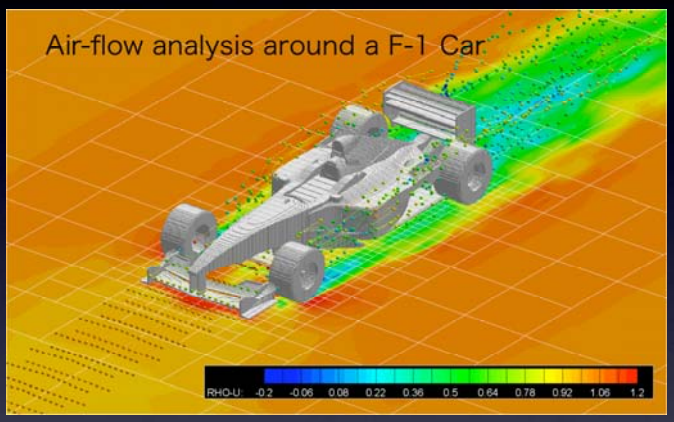
```
do km = 1, nd
   do jq = 1, nsum_dip
      do iq = 1, nsum_dip
           sum1(iq,km) = sum1(iq,km)+ r(jq,km) * gd(iq,jq,km)
           sum2(iq,km) = sum2(iq,km)+ i(jq,km) * gd(iq,jq,km)
      end do
   end do
end do
```

Unrolling

```
do km = 1, nd
 do jq = 1, nsum_dip, k
  do iq = 1, nsum_dip
   sum1(iq,km) = sum1(iq,km)+r(jq,km)*gd(iq,jq,km)
    +r(jq+1,km)* gd(iq,jq+1,km)+r(jq+2,km)* gd(iq,jq+2,km)
     ...
    +r(jq+k-1,km)*gd(iq,jq+k-1,km)
   sum2(iq,km) = sum2(iq,km)+ i(jq,km)*gd(iq,jq,km)
    + i(jq+1,km)*gd(iq,jq+1,km)+ i(jq+2,km)*gd(iq,jq+2,km)
     ...
    + i(jq+k-1,km)* gd(iq,jq+k-1,km)
  end do
 end do
end do
```



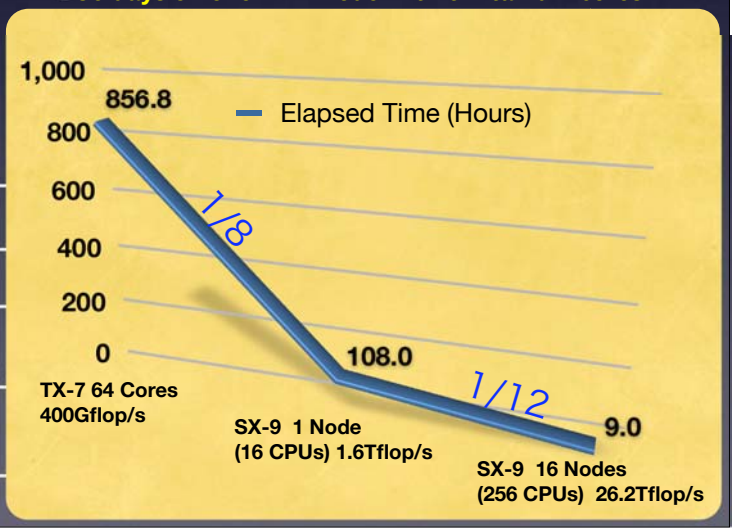
16-Node Performance in CFD



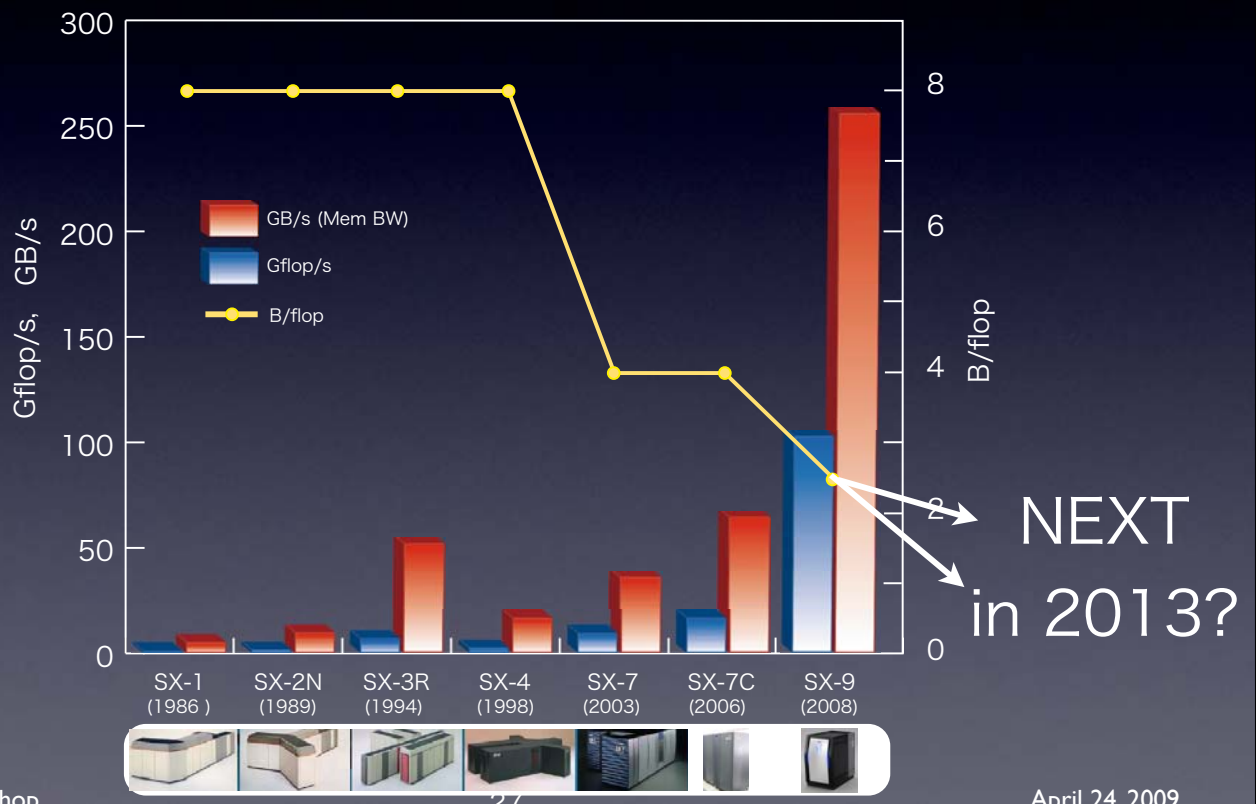
- Started with a scalar-tuned code for TX-7/i9610
- Almost 99.9 % vector performance was achieved.
- 0.2 billion cells were solved by present method.
- Flat MPI shows better parallel efficiency than hybrid.
- 161x speedup obtained on the 16 nodes with 256 CPUs
- 9 hours on 16 nodes (256 CPU) of SX-9
- 36 days on one TX-7 node with 64 itanium cores

Comparison with a TX-7 scalar system

	TX7(Itanium)		SX-9		
Cores	1	64	1	16	256
Peak Perf.	6.4GF (1x)	409.GF (64x)	102.G (16x)	1.6TF (256x)	26TF (4096x)
Sustained Speedup	1x	36x	21x	316x	3460x

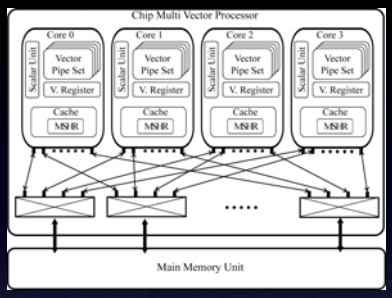


SX Performance Trend



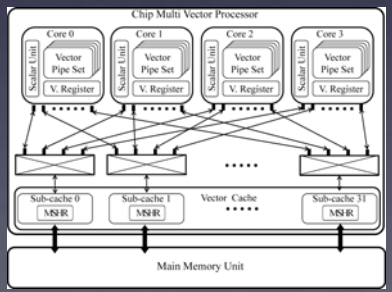
Design Choices for the Next Vector Processor

- Multicore
  - ✓ increasing flop/s rate
  - ✓ SMP on a chip
  - ✻ Limited Memory Bandwidth
    - decreasing B/F rate per core
- Large on-chip cache
  - ✓ decreasing off-chip memory access
  - ✓ decreasing IO driving power
  - private
    - ✓ exclusive, no conflict
    - ✓ fast
    - ✻ limited capacity
  - shared
    - ✓ large
    - ✓ effective for shared data in SMT
    - ✻ access conflicts
    - ✻ limited B/F rate
  - distributed shared
  - multi-level
    - ✓ 1st-level fast private
    - ✓ 2nd-level large shared



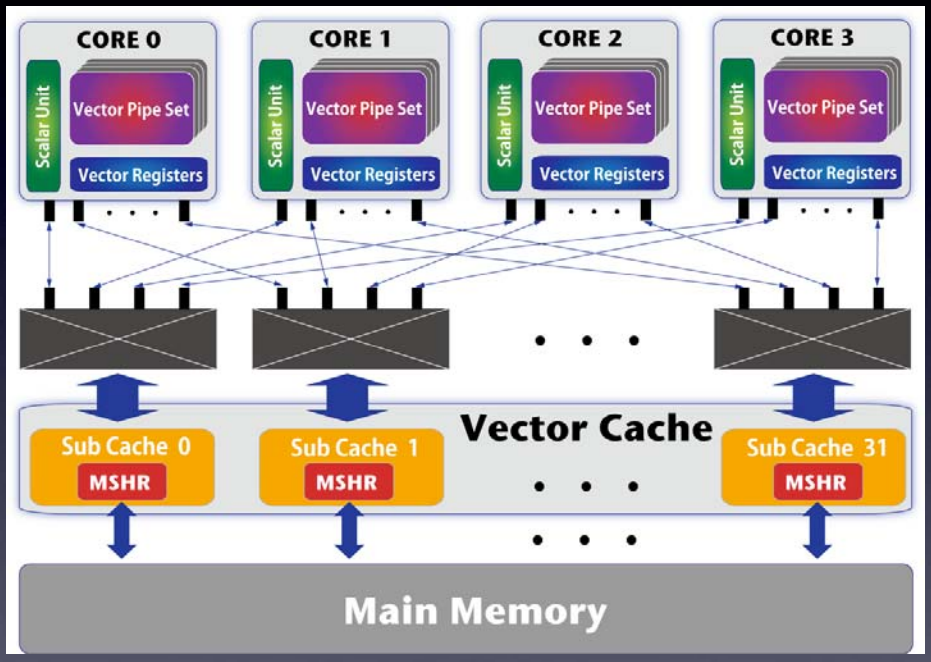
Fully-Distributed (private)

Hybrid
(Distributed shared/Multi-level)



Centralized (Shared)

Toward a Multi Vector Core Processor!



- CMVP
  - ✓ 4 cores
  - ✓ Shared cache
- Core
  - ✓ NEC SX architecture
- Shared vector cache
  - ✓ 32 sub-caches
  - ✓ 2-way set-associative
  - ✓ Write-through
  - ✓ 8B line sizes
  - ✓ MSHR

A.Musa, Y.Sato, T.Soga, R. Egawa, H. Takizawa,H. Kobayashi, "Caching for A Chip Multi Vector Processor," presented at SC08, 2008.

Prefetching Effects of the On-chip Shared Vector Cache in Multithreading of the Difference Scheme

FDTD kernel

```
DO 10 k=0,Nz ; DO 10 i=0,Nx; DO 10 j=0,Ny
        E_x(i,j,k) = C_x_a(i,j,k)*E_x(i,j,k)
&               + C_x_b(i,j,k) * (( H_z(i,j,k) - H_z(i,j-1,k))/dy
&               - (H_y(i,j,k) - H_y(i,j,k-1))/dz - E_x_Current(i,j,k))
        E_z(i,j,k) = C_z_a(i,j,k)*E_z(i,j,k)
&               + C_z_b(i,j,k) * ((H_y(i,j,k)-H_y(i-1,j,k) )/dx
&               - (H_x(i,j,k)-H_x(i,j-1,k) )/dy - E_z_Current(i,j,k))
        E_y(i,j,k) = C_y_a(i,j,k)*E_y(i,j,k)
&               + C_y_b(i,j,k) * ((H_x(i,j,k) - H_x(i,j,k-1) )/dz
&               - (H_z(i,j,k) - H_z(i-1,j,k))/dx - E_y_Current(i,j,k))
10 CONTINU
```

inter-thread locality

intra-thread locality

inter-thread locality

intra-thread locality

A.Musa, Y.Sato, T.Soga, R. Egawa, H. Takizawa,H. Kobayashi, "Caching for A Chip Multi Vector Processor," presented at SC08, 2008.

Thread Mapping of Difference Code on Cores

```
DO 10 k=0,Nz
  DO 10 i=0,Ny
    DO 10 j=0,Nx
      ~ = ~ (H_y(i,j,k) – H_y(i,j,k-1)) ~
10 CONTINUE
```

Core 0

```
DO 10 k=0,Nz,4
  DO 10 j=0,Ny
    DO 10 i=0,Nx
      ~ = ~ (H_y(i,j,k) – H_y(i,j,k-1)) ~
10 CONTINUE
```

Core 1

```
DO 10 k=1,Nz,4
  DO 10 j=0,Ny
    DO 10 i=0,Nx
      ~ = ~ (H_y(i,j,k) – H_y(i,j,k-1)) ~
10 CONTINUE
```

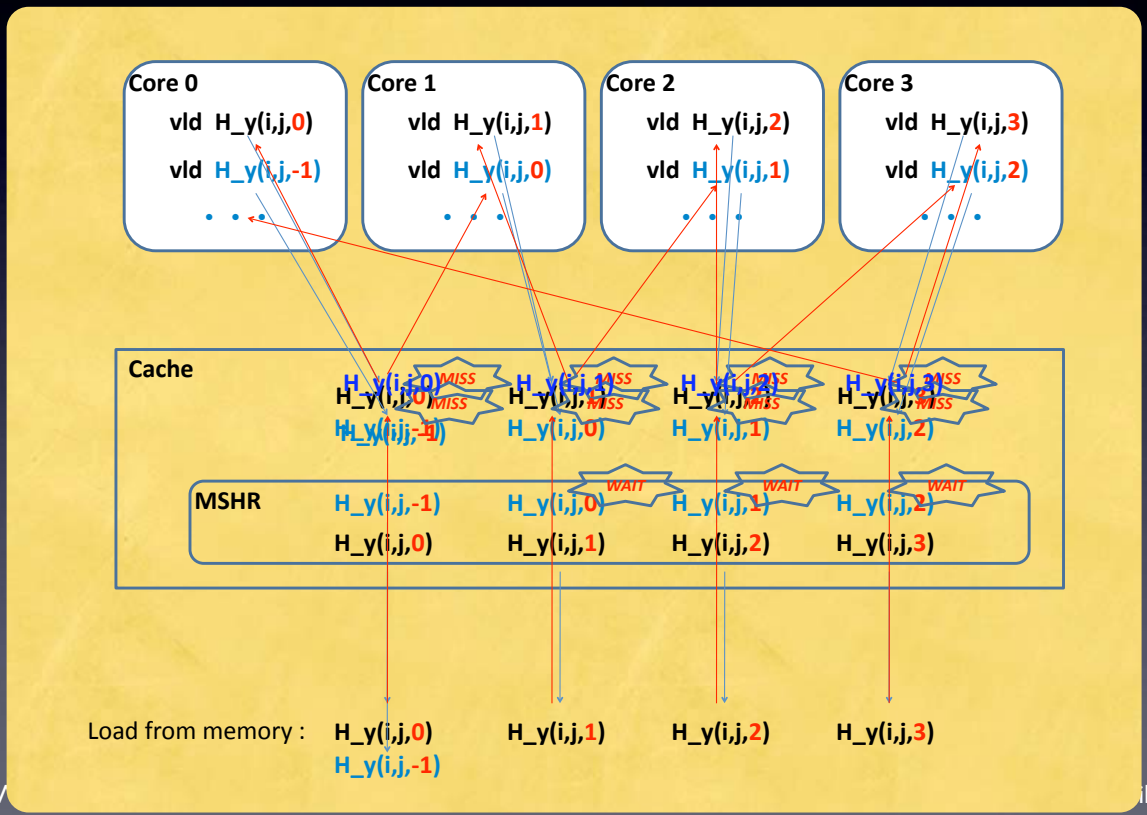
Core 2

```
DO 10 k=2,Nz,4
  DO 10 j=0,Ny
    DO 10 i=0,Nx
      ~ = ~ (H_y(i,j,k) – H_y(i,j,k-1)) ~
10 CONTINUE
```

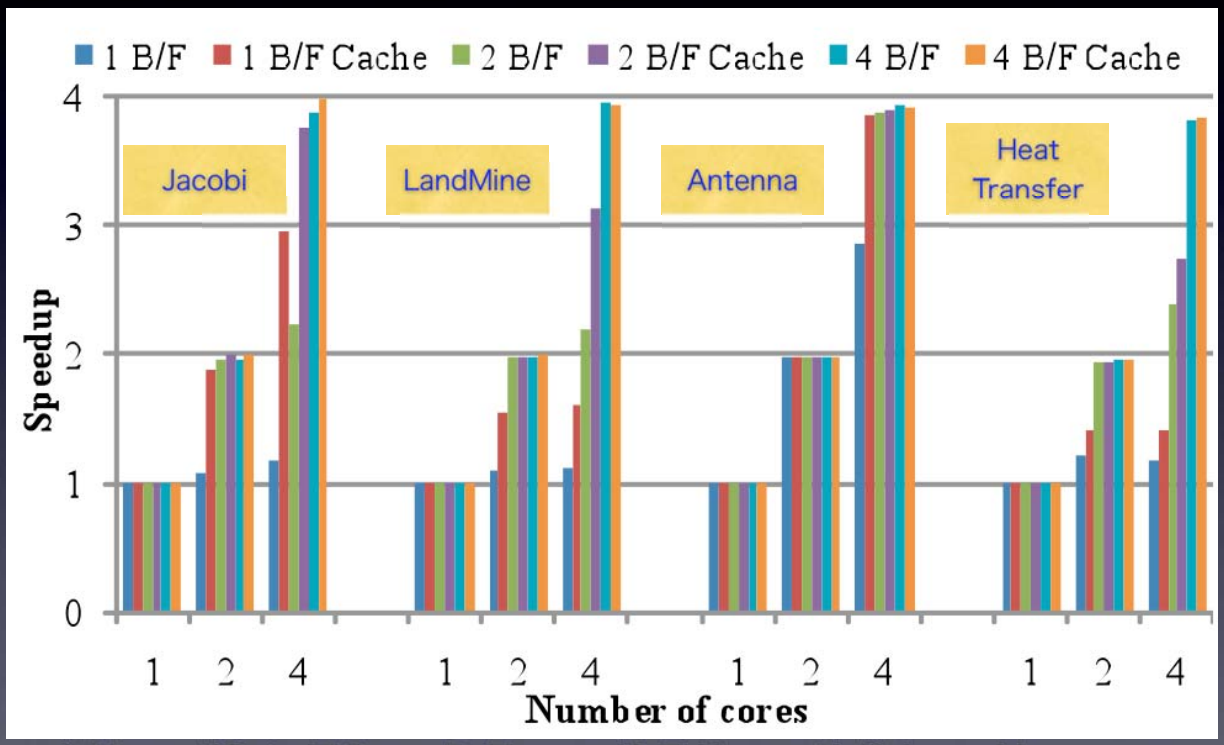
Core 3

```
DO 10 k=3,Nz,4
  DO 10 j=0,Ny
    DO 10 i=0,Nx
      ~ = ~ (H_y(i,j,k) – H_y(i,j,k-1)) ~
10 CONTINUE
```


Cache Behavior on Cores

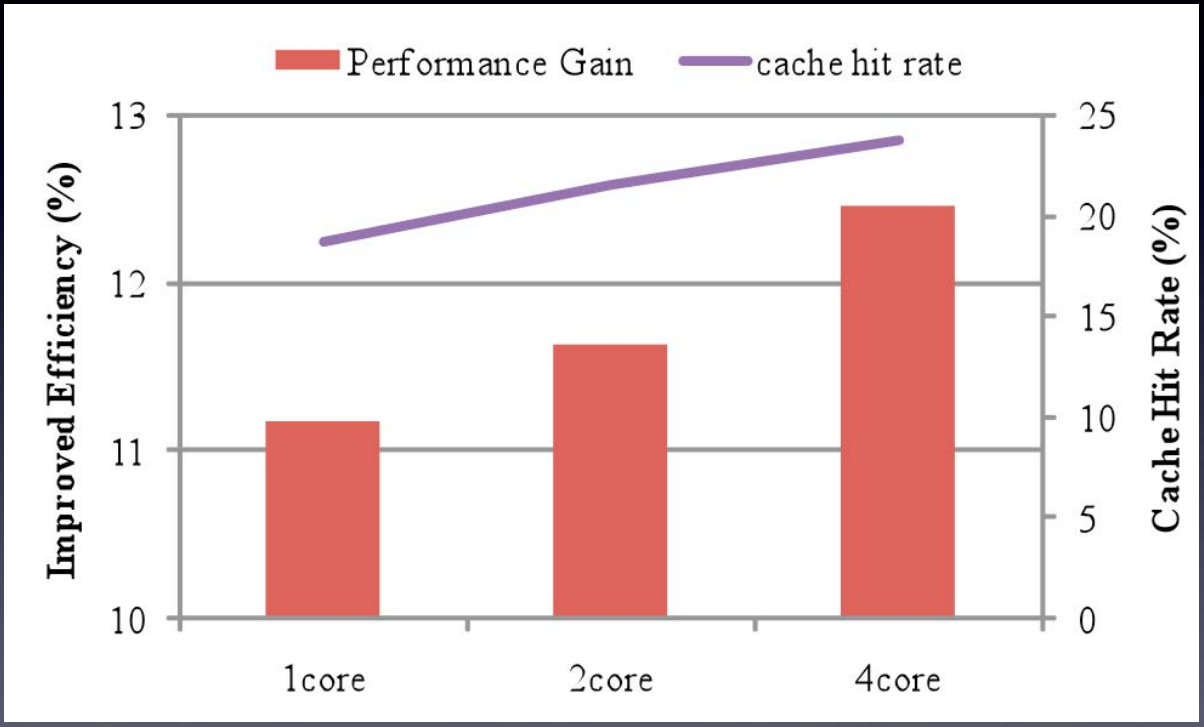


Performance of Multi Vector-Cores with the Shared Cache



A.Musa, Y.Sato, T.Soga, R. Egawa, H. Takizawa, H. Kobayashi, "Caching for A Chip Multi Vector Processor," presented at SC08, 2008.

Prefetching Effects of the On-chip Shared Vector Cache in Multithreading



A.Musa, Y.Sato, T.Soga, R. Egawa, H. Takizawa, H. Kobayashi, "Caching for A Chip Multi Vector Processor," presented at SC08, 2008.

Lessons learned from SX-9 Experiences and Towards the Next-Generation Vector Computing

- Great potentials of SX-9
  - Powerful tool for "Short Time to Innovations" in computational science
    - 19 top one scores on 28 HPCC benchmark tests!
- The first on-chip cache mechanism for the SX architecture works well!
  - Definitely covers the lack of off-chip memory bandwidth, but...
  - more capacity, more sophisticated data management needed
- Towards the Next-Generation Vector Computing
  - Virtualization of distributed vector computing resources
  - Multicore desing of the vector architecture
    - Research challenges
      - Hardware/software-controlled optimizations for on-chip data handling needed
        - Tradeoff between loop-unrolling & selective caching with prefetching, outstanding load handling on miss
      - New memory hierarchy design for a multicore era of the vector architecture under the consideration of power consumption and sustained performance

Hiroaki Kobayashi, Tohoku University

Acknowledgements

- Tohoku University
  - Koki Okabe
  - Ryusuke Egawa
  - Hiroyuki Takizawa
  - Ei-ichi Ito
  - Kenji Oizumi
  - Other colleagues and students of the project
- NEC
  - Akihiko Musa
  - Takashi Soga
  - Youichi Shimomura
  - Yoko Isobe
  - Tatsunobu Kokubo
  - Naoyuki Sogo
  - Masaaki Yamagata
  - Other NEC Engineers involved in SX R&D

Disclaimer:Information provided in this talk does not reflect any future design of the NEC systems.