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.
Lessons Learned from 1-year SX-9 Experience and Toward the Next Generation Vector Computing

Hiroaki Kobayashi
Director and Professor
Cyberscience Center, Tohoku University
koba@isc.tohoku.ac.jp

20th CCSE Workshop
April 24, 2009

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

**Features of SX-9 of Tohoku Univ.**

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

<table>
<thead>
<tr>
<th></th>
<th>SX-7 in 2003</th>
<th>SX-9 in 2008</th>
<th>improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Per CPU</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Freq.</td>
<td>1.1GHz</td>
<td>3.2GHz</td>
<td>2.9x</td>
</tr>
<tr>
<td>Vec. Perf.</td>
<td>8.8Gflop/s</td>
<td>102.4Gflop/s</td>
<td>11.6x</td>
</tr>
<tr>
<td>Mem. BW</td>
<td>35.3Gflop/s</td>
<td>256Gflop/s</td>
<td>7.3x</td>
</tr>
<tr>
<td>Vec. Perf.</td>
<td>282Gflop/s</td>
<td>1.6Tflop/s</td>
<td>5.8x</td>
</tr>
<tr>
<td>Mem. Cap.</td>
<td>256GB</td>
<td>1TB</td>
<td>4x</td>
</tr>
<tr>
<td>Mem. BW</td>
<td>1.3TB/s</td>
<td>4TB/s</td>
<td>3.5x</td>
</tr>
<tr>
<td>Mem. Banks</td>
<td>16K</td>
<td>32K</td>
<td>2.0x</td>
</tr>
<tr>
<td>IXS BW</td>
<td>32G/s*</td>
<td>256Gflop/s</td>
<td>8.0x</td>
</tr>
<tr>
<td><strong>Per SMP Node</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>System</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Perf.</td>
<td>2.1Tflop/s</td>
<td>26.2Tflop/s</td>
<td>12.5x</td>
</tr>
<tr>
<td>Total Mem.</td>
<td>2TB</td>
<td>16TB</td>
<td>8x</td>
</tr>
</tbody>
</table>

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

TX7/i9610 (3-node) 1.23Tflop/s, 1.5TB (Installed in 2006)
SX-9 Processor Architecture

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

102.4 = 4 ops. x 8 units x 3.2 GHz

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

Top One Scores in 28 Tests (as of 2008.11.18)

System Specifications

<table>
<thead>
<tr>
<th>System name</th>
<th>manufacture</th>
<th>Processor Type</th>
<th>Freq.</th>
<th># of Cores</th>
<th># of MPI Proc</th>
<th># of Threads</th>
<th>Peak [TF]</th>
<th>Interconnect</th>
<th>Network BW</th>
</tr>
</thead>
<tbody>
<tr>
<td>SX-9</td>
<td>NEC</td>
<td>SX-9</td>
<td>3.2GHz</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>26.2</td>
<td>IXS</td>
<td>128GB/s</td>
</tr>
<tr>
<td>SX-9(SMP)</td>
<td>NEC</td>
<td>SX-9</td>
<td>3.2GHz</td>
<td>32</td>
<td>2</td>
<td>16</td>
<td>3.2</td>
<td>IXS</td>
<td>128GB/s</td>
</tr>
<tr>
<td>SX-8</td>
<td>NEC</td>
<td>SX-8</td>
<td>2GHz</td>
<td>40</td>
<td>40</td>
<td>1</td>
<td>0.64</td>
<td>IXS</td>
<td>16GB/s</td>
</tr>
<tr>
<td>SX-8(SMP)</td>
<td>NEC</td>
<td>SX-8</td>
<td>2GHz</td>
<td>40</td>
<td>5</td>
<td>8</td>
<td>0.64</td>
<td>IXS</td>
<td>16GB/s</td>
</tr>
<tr>
<td>SX-7</td>
<td>NEC</td>
<td>SX-7</td>
<td>0.552GHz</td>
<td>32</td>
<td>32</td>
<td>1</td>
<td>0.28256</td>
<td>non</td>
<td>non</td>
</tr>
<tr>
<td>SX-7(SMP)</td>
<td>NEC</td>
<td>SX-7</td>
<td>0.552GHz</td>
<td>32</td>
<td>2</td>
<td>16</td>
<td>0.28256</td>
<td>non</td>
<td>non</td>
</tr>
<tr>
<td>Idataplex</td>
<td>IBM–Serviware</td>
<td>Xeon X5472</td>
<td>3.0GHz</td>
<td>1088</td>
<td>1,088</td>
<td>1</td>
<td>13.5</td>
<td>Infiniband</td>
<td>2GB/s</td>
</tr>
<tr>
<td>BL2x220</td>
<td>HP</td>
<td>Xeon E5450</td>
<td>3.0GHz</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>3.072</td>
<td>Infiniband</td>
<td>2GB/s</td>
</tr>
<tr>
<td>SCS832</td>
<td>SiCortex</td>
<td>SiCortex Ice9</td>
<td>0.7GHz</td>
<td>5760</td>
<td>5,760</td>
<td>1</td>
<td>8.064</td>
<td>Custom</td>
<td>6GB/s</td>
</tr>
<tr>
<td>Blue Gene/P</td>
<td>IBM</td>
<td>PowePC450</td>
<td>0.85GHz</td>
<td>131,072</td>
<td>131,072</td>
<td>1</td>
<td>557</td>
<td>Torus</td>
<td>425MB/s</td>
</tr>
<tr>
<td>Blue Gene/P (SMP)</td>
<td>IBM</td>
<td>PowePC450</td>
<td>0.85GHz</td>
<td>131,072</td>
<td>32,768</td>
<td>4</td>
<td>557</td>
<td>Torus</td>
<td>425MB/s</td>
</tr>
<tr>
<td>XT5</td>
<td>CRAY</td>
<td>AMD Opteron</td>
<td>2.3GHz</td>
<td>149,058</td>
<td>74,529</td>
<td>2</td>
<td>1,381.62</td>
<td>SeaStar</td>
<td>9.6GB/s</td>
</tr>
<tr>
<td>Darwin</td>
<td>ClusterVision/Dell/QLogic</td>
<td>Xeon 5160</td>
<td>3GHz</td>
<td>256</td>
<td>256</td>
<td>1</td>
<td>3.072</td>
<td>Infiniband</td>
<td>2GB/s</td>
</tr>
<tr>
<td>Altix 8200EX</td>
<td>SGI</td>
<td>Xeon X5472</td>
<td>3GHz</td>
<td>1,024</td>
<td>1,024</td>
<td>1</td>
<td>12.288</td>
<td>Infiniband</td>
<td>2GB/s</td>
</tr>
<tr>
<td>Intel Endeavor cluster</td>
<td>Intel</td>
<td>Xeon 5160</td>
<td>3GHz</td>
<td>1,024</td>
<td>1,024</td>
<td>1</td>
<td>11.4688</td>
<td>Infiniband</td>
<td>2GB/s</td>
</tr>
</tbody>
</table>

20th CCSE Workshop

Hiroaki Kobayashi, Tohoku University

April 24, 2009
STREAM (Averaged)

Random Access (SN/EP)
**G-FFT Results: Highly Efficient Computing on SX-9**

(As of 2008.11.18)

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

![Latency Chart]

Bandwidth

![Bandwidth Chart]
PingPong Performance of SX-9 Ixs

- Peak: 128 GB/s in each direction

- Sustained BW (GB/s)
  - 0: 29.55GB/s (23%) on 2MB (HPCC Specified)
  - 8: 32 GB/s
  - 96: 101.09GB/s (79%) on 128MB
  - 160: 103.03GB/s (89%) on 256MB

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
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

```fortran
!cdir on adb(dvecw).nodep
do ii=jj,min(jj+i流传1.i plast)
    kk = ii * jj + 1
    aa = xpl1(ii)/delx
    ic = int(aa+half)
    raal = ic
    dd1 = fact1*(aa+half-raal)
    dd2 = fact1*(raal-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

```fortran
sum(1,1,k) & sum(1,2,k) have temporal locality but they need 1.6MB cache space!
sum(i,1,k) = sum(i,1,k) + dble(a(i,k))*b(i,1,j) + aimag(a(i,k))*b(i,2,j)
sum(i,2,k) = sum(i,2,k) + dble(a(i,k))*b(i,2,j) + aimag(a(i,k))*b(i,1,j)
end do
```

Selective caching with blocking

```fortran
vlen : cache blocking parameter (if vlen <= 12,890, sum(1,1,k) & sum(1,2,k) are cacheable on the 256KB ADB)
```

```fortran
do k=1,l
    do j=1,m
        sum(1,1,k) = sum(1,1,k) + dble(a(i,k))*b(i,1,j) - aimag(a(i,k))*b(i,2,j)
        sum(1,2,k) = sum(1,2,k) + dble(a(i,k))*b(i,2,j) - aimag(a(i,k))*b(i,1,j)
    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 Icdir ON_ADB(H_x)
04 Icdir ON_ADB(H_y)
05 Icdir 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 & + C_x_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 & + C_y_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

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 Icdir ON_ADB(H_x)
04 Icdir ON_ADB(H_y)
05 Icdir 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

```fortran
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

```fortran
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

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

Unrolling

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

Air-flow analysis around a F-1 Car

Comparison with a TX-7 scalar system

<table>
<thead>
<tr>
<th></th>
<th>TX7 (Itanium)</th>
<th>SX-9</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cores</td>
<td>1</td>
<td>64</td>
</tr>
<tr>
<td>Peak Perf.</td>
<td>6.4GF (1x)</td>
<td>409 GF (64x)</td>
</tr>
<tr>
<td></td>
<td>102.6 GF (16x)</td>
<td>1.6 TF (256x)</td>
</tr>
<tr>
<td>Sustained Speedup</td>
<td>1x</td>
<td>36x</td>
</tr>
<tr>
<td></td>
<td>21x</td>
<td>316x</td>
</tr>
<tr>
<td></td>
<td>3460x</td>
<td></td>
</tr>
</tbody>
</table>

Towards Next Generation Vector Computing

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

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-1,j,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-1,j,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 CONTINUE
```

A.Musa, Y.Sato, T.Soga, R. Egawa, H. Takizawa, H. Kobayashi,
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

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

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

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

---

Cache Behavior on Cores

![Cache Diagram](image)

- Core 0: vld H_y(i,j,0), vld H_y(i,j,-1)
- Core 1: vld H_y(i,j,1)
- Core 2: vld H_y(i,j,2)
- Core 3: vld H_y(i,j,3)

- MSHR: H_y(i,j,0), H_y(i,j,1), H_y(i,j,2), H_y(i,j,3)
- Load from memory: H_y(i,j,0), H_y(i,j,1), H_y(i,j,2), H_y(i,j,3)
Performance of Multi Vector-Cores with the Shared Cache

![Graph showing speedup with different numbers of cores and cache sizes.]

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

![Graph showing performance gain and cache hit rate with different numbers of cores.]
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 design 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

**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.