Single address space facilitates a simple programming model

(Cost: UMA or NUMA)

All processors share a single global address space

Physically shared address space machines
Cost

Scalability

Performance - Programming Techniques

These aspects determine:

Cache coherence mechanism

Interconnection

Memory organization (physically shared)
Banks service read/write requests independently
to banks

Non-overlapping regions of address space mapped

Split memory across multiple modules (banks)

Memory Interleaving

Causes sequentialization of accesses

Single memory module shared among processors

Memory Organization
high rate of bank conflicts

- Block transfer possible only at bank rate and
- High-order bits of address used to select bank

High-order interleaving •

- Enables block transfers and reduces bank conflicts
- Low-order bits of address used to select bank

Low-order interleaving •
Reordering to fit

- Consider column vs. row access and loop

Restructuring

- Involves data placement as well as code

- Must spread accesses across banks to avoid bank conflicts

Programming issues

- Conflicts

   - Row-order interleave is used because it reduces bank

   - Typically, in tightly coupled parts of system

Memory Organization
A simple linear interleaving of addresses on an eight-bank memory system is shown below.

<table>
<thead>
<tr>
<th>B0</th>
<th>B1</th>
<th>B2</th>
<th>B3</th>
<th>B4</th>
<th>B5</th>
<th>B6</th>
<th>B7</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>14</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>10</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
processor and/or across a network to avoid congestion.

Memory traffic across banks that are slower than the others are possible. This one is typically used to spread the therefore based on the low-order bits of the address. Many

where \( B \) is the number of banks used. The interleaving is

\[ B \mod u = u \text{ if address } A \text{ is given by } A = B \mod u \]

bank number, \( u \), or address \( A \) is assigned to bank \( 0 \), the

Assuming that memory address 0 is assigned to bank 0, the
for servicing on a bank with a cycle busy time. May have to wait 1 to 1 cycles before being accepted. The second bank which is servicing a previous request, the second memory request arrives at a memory access.

Bank busy time = amount of time a bank takes to service a memory access on a vector processor.

Degradation on a vector processor is one of the key sources of performance variables (or other structured elements of a vector or vectors) or other structured contention for memory banks by accesses of different
number of memory banks.

\[ \frac{\gcd(B', \rho)}{\gcd(B', \alpha)} = \alpha \]

where \( \alpha \) is the greatest common divisor of \( \alpha \) and \( \beta \), \( \beta \) is the some bank, then the next address that is in the same bank is which is generated with stride \( \rho \), i.e., \( \alpha + \rho \). If \( \rho \) is in Let \( \alpha \), \( \beta \) = 1, \ldots, \in be a sequence of addresses for some vector

Detecting conflicts
as desired.

\[ \frac{(B', \sigma) e \cdot d, c, B}{B} = a \leftrightarrow \frac{(B', \sigma) e \cdot d, c, B}{I} = \frac{z_\mu}{I_\mu} \]

Finally we have \( z_\mu = e \cdot d, c, B \). It follows that

For any \( l \) choosing \( z_\mu = e \cdot d, c, B \) minimizes \( a \).

With \( \sigma \) this implies that \( \tau_2 \) is a common divisor of \( I_\mu \) and \( I_\mu \).

where \( g \) is an integer. Therefore, \( \sigma \) which implies that \( \tau_2 \) is a divisor of

\[ \sigma \frac{\tau_2}{I_\mu} = \sigma = B \cdot \]

We also have

\( \sigma \) as an integer \( \tau_2 \) implies that \( \tau_2 \) is a divisor of \( I_\mu \).

where \( I \) \( \mu \) \( \tau_2 \) are integers.

\[ \frac{\tau_2}{I_\mu} = a \]

\( \sigma \) as an integer we know that

\( \frac{B}{\sigma} \) as an integer \( B \) and we know that \( I \) \( \mu \) \( \sigma \) are in the same bank. So the longest burst of stride \( \sigma \) accesses that can be in

and \( a + B \) are in the same bank. So the longest burst of stride \( a \) accesses that can be in

We want the smallest \( a \) such that \( (a \sigma) \) mod \( B = 0 \). It is easy to see that if \( a \) then \( a \).
Table:

<table>
<thead>
<tr>
<th>B^7</th>
<th>B^6</th>
<th>B^5</th>
<th>B^4</th>
<th>B^3</th>
<th>B^2</th>
<th>B^1</th>
<th>B^0</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_4$</td>
<td>$q_3$</td>
<td>$q_2$</td>
<td>$q_1$</td>
<td>$q_0$</td>
<td>$q_3$</td>
<td>$q_2$</td>
<td>$q_1$</td>
</tr>
</tbody>
</table>

Example: Dimension A(8, 4\(^t\), B(4, 4\(^t\))

addresses

Portray uses column major ordering to map arrays to
and references to rows of each array all go to the same bank.

Therefore, \( a = 1 \), i.e., accesses are at multiples of 8 addresses.

\[
0 = \text{leading dimension} \times \text{increment} = 20 \times 2 = 40
\]

\[
\text{do end}
\]

\[
\text{do end}
\]

\[
(\zeta, \xi, \eta) \times (\zeta, \xi, \eta) + (\xi, \eta, \xi) \times (\xi, \eta, \xi) = (\xi, \eta, \xi)
\]

\[
\text{do } j = 1, 20, 2 \text{ do}
\]

\[
\text{do } i = 1, 20, 20
\]

\[
\text{DIMENSION } x(20, 20), y(20, 20), z(20, 20)
\]

\text{Example:}
same row or column now map to different consecutive banks

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a₃,4</td>
<td>a₄,4</td>
<td>a₃,4</td>
<td>a₂,4</td>
<td>a₁,4</td>
<td>a₉,3</td>
<td>a₈,3</td>
<td>a₇,3</td>
<td>a₆,3</td>
<td></td>
</tr>
<tr>
<td>a₆,3</td>
<td>a₅,3</td>
<td>a₄,3</td>
<td>a₃,3</td>
<td>a₂,3</td>
<td>a₁,3</td>
<td>a₉,2</td>
<td>a₈,2</td>
<td>a₇,2</td>
<td></td>
</tr>
<tr>
<td>a₇,2</td>
<td>a₆,2</td>
<td>a₅,2</td>
<td>a₄,2</td>
<td>a₃,2</td>
<td>a₂,2</td>
<td>a₁,2</td>
<td>a₉,1</td>
<td>a₈,1</td>
<td></td>
</tr>
<tr>
<td>a₈,1</td>
<td>a₇,1</td>
<td>a₆,1</td>
<td>a₅,1</td>
<td>a₄,1</td>
<td>a₃,1</td>
<td>a₂,1</td>
<td>a₁,1</td>
<td>a₀,0</td>
<td></td>
</tr>
</tbody>
</table>

Example: dimension A(9, 4)

Dimensions for arrays.

For systems with an even number of banks, use odd indexing to reduce memory conflicts.
Example:

Conflicts between two data arrays, No COMMON blocks with dummy elements to reduce

common /x(2,1024), y(2,1024)/

end do
end do

0 = x(1,1) - 1.0

do j = 1, 1024

do i = 1, 2

Example:
Then the address in the rows of $y$ are shifted one bank and there is no conflict between the $i$-th row of $x$ and the $i$-th row of $y$. 
Now the even elements are located in
proceeded in the same way – combine odd with even and
On the second pass, the result vector of first stage is
banks, i.e., at full rate.
access of the two vectors is serviced by all of the memory
Note each operand is stride two but the combined
result vector of length \( n/2 \). The result overwrites the even
combined with the odd elements of the vector to produce a
On the first pass, the even elements of the vector are
number of banks.
laid out in an interleaved memory system with an even
Suppose we have an algorithm running on a vector processor

Data Reshaping to Mitigate Conflicts
On subsequent passes, the stride doubles and fewer memory banks are used. 

4 and the overall is stride 2 and only half the banks are used. Therefore each are stride 4 + \( \frac{x}{4} \) (and the odds in \( A(\frac{x}{2} + \frac{4}{4}) \)).
\[ \log_2 n + \frac{1}{4} \log_2 n + \frac{1}{2} \log_2 n + 1 \approx 2n. \]

So the total space required for the first step is \( \frac{1}{4} \log_2 n \) needed for the first step, \( \frac{1}{2} \log_2 n \) needed for the second step, \( \log_2 n \) needed for the result, and \( 1 \) location (or \( 7 \) locations) padded on the result to \( \log_2 n \) locations padded on to the initial \( n \). On the first step rather than overwriting one of the vectors, write the cost is doubling the space required for the algorithm. On each access on independent banks.

This problem can be fixed by reshaping the arrays so that the combined accesses are to data with stride 1, i.e., two stride \( 2 \).
- Relationship between parameters of stream – relative
  
  simultaneousty (multiple ports)
  
  between streams (one port), multiple streams
  short lengths due to registers and loop control, switch
  burst of mapping of stream template to architecture – burst of
  
  S, I, S, ILS with starting address, strides, lengths
  number of vector streams involved and their type – T.
  
  Determine the mode of access model

  space to the memory banks.

  Determine the mapping of the linear physical address
  structure to the linear physical address space.

  Determine the mapping of the multidimensional data

  Key tasks in interleave-consecutive coding
Dynamic reshaping is possible.

Patterns occur increase scope of optimization and/or use

When conflicts between optimal access and store

across multiple primitives

Mitigate if possible and necessary within one primitive

Determine effective fraction of memory bandwidth used.

Streams, relationship between strides,

offset of base addresses, pattern of switching between
Caches are used to exploit temporal locality in programs.
Processor

Memory

Access = 10 cycles

Cache

Access Level -2 cache = 4 cycles

Access Register = 1 cycle

Access L1 cache = 2 cycles

Bus
Don't update block in main memory - write-back

- Update block in main memory - write-through

- On write to a block at a processor:
  - write-update

- Update copies of the block for other processors
  - write-invalidate

- Invalidate copies of the block for other processors
  - On write to a block at a processor:

- Cache blocks

- Needed to avoid incorrect results due to sharing of

Cache Coherence Protocols
Types of cache coherence mechanisms:

- Directory-based cache coherence protocols
  - Snoopy cache coherence protocols
Write-back versus write-through

Cache Coherence Protocols
Bus
Memory

Caches

Processors

(Write-through)

(Write-back)
Cache Coherence Protocols

- Write-invalidate versus write-update
Write-invalidate protocol for a write-through cache

• Can use state-transition graphs to represent protocols

affecting their current cache blocks

• All processors monitor bus traffic for information

Snoop Cache Coherence Protocols
memory module

- Distributed - information for cache blocks in each
  together

- Centralized - information for all cache blocks

**Directory organization:**

- Locate latest version of a cache block
- Access to memory through the directory helps cache block

**Directory based schemes track processors using each:**

- Directory based schemes track processors using each
- Precious bandwidth - bus systems are an exception
- Snoopy protocols involve repeated broadcasts using

**Snoopy Protocols**

**Directory-based Cache Coherence Protocols**
- Amdahl's law applies to mix
  replacement policy crucial
  used several times in a short period of time
  When a block is fetched from memory, it will be
  Temporal locality

  similar to the idea of a page in virtual memory
  depending on where it appears in memory system
  cache lines used - from 4 words to over 100
  elements must be used
  When a block is fetched from memory, all
  Spatial locality

  Effect of Caches on Performance
SYNCHRONIZATION, it is necessary not sufficient

COHERENCE DOES NOT GUARANTEE •

- Block ping-pongs between processors

- Same block

- Processors write to non-overlapping parts of the

False sharing •

Effect of Caches on Performance
and message-passing programming

Programming supports data parallel, work sharing

(150 MHz)

Clock is synchronized across system for 6.67 ms period

access data in remote processor's memory

Hardware: Processors can use `put`, `get` primitives to

No cache coherence among remote memory in

Gray T3D
Cray T3D Overview
network interface, and a block transfer element.

Each PE node contains two processing elements, a

support circuitry

Each processing element contains processor memory.

Cray T3D Processing Element
Cray T3D Interconnect

- Peak communication bandwidth: 300 MB/sec
- Interconnect is a 3-D torus connection
- Each direction: 76.8 GB/sec bisection bandwidth
I/O Gateway
To Host
Interconnect
Processing Element
Node
System
Host
I/O Gateway

+Y
+X
+Z
-Y
-X
-Z
Q
R
ST
Q
S
U
T
W
Y
X
Z
V
W
X
S
T
W
Y
Z
 µ
d
z
abc
`