Some programmers don’t like the fact that distributed memory machines require programming using message passing and prefer the shared address space model: shared memory is easier to write programs for (they claim)
Some programmers don’t like the fact that distributed memory machines require programming using message passing and prefer the shared address space model: shared memory is easier to write programs for (they claim)

They can use virtual shared memory
Classifications
Virtual Shared/Distributed Virtual Memory

Some programmers don’t like the fact that distributed memory machines require programming using message passing and prefer the shared address space model: shared memory is easier to write programs for (they claim)

They can use *virtual shared memory*

Just as virtual memory is a way of converting virtual memory addresses into physical memory addresses, virtual shared memory is a mechanism to have a single, virtual, address space that is converted into distributed physical addresses
Some programmers don’t like the fact that distributed memory machines require programming using message passing and prefer the shared address space model: shared memory is easier to write programs for (they claim)

They can use *virtual shared memory*

Just as virtual memory is a way of converting virtual memory addresses into physical memory addresses, virtual shared memory is a mechanism to have a single, virtual, address space that is converted into distributed physical addresses

Thus this is also called *distributed virtual memory* and *distributed shared memory*
Classifications
Virtual Shared/Distributed Virtual Memory

Reading and writing data will be very NUMA and probably implemented by a message passing layer hidden from the programmer in the OS or systems libraries
Reading and writing data will be very NUMA and probably implemented by a message passing layer hidden from the programmer in the OS or systems libraries.

So the programmer won’t have to care about it and they can write programs as if the whole of memory was one big chunk.
Classifications
Virtual Shared/Distributed Virtual Memory

Reading and writing data will be very NUMA and probably implemented by a message passing layer hidden from the programmer in the OS or systems libraries.

So the programmer won’t have to care about it and they can write programs as if the whole of memory was one big chunk.

The programmer writes “\(x = y\)” and the compiler/OS converts this into a shared memory access or a message call as appropriate.
Unfortunately, programmers do have to care as the speed of a program will be very hard to predict or control, depending on how data is distributed across memory.
Unfortunately, programmers *do* have to care as the speed of a program will be very hard to predict or control, depending on how data is distributed across memory.

E.g., how long does the assignment “\(x = y\)” take?
Unfortunately, programmers do have to care as the speed of a program will be very hard to predict or control, depending on how data is distributed across memory.

E.g., how long does the assignment “\(x = y\)” take?

This would strongly affect how we would employ variables in a program.
Classifications
Virtual Shared/Distributed Virtual Memory

The underlying system also needs to solve all the problems of cache coherence that shared memory hardware has, but now using the (relatively) slow messaging passing layer rather than custom-designed hardware.
The underlying system also needs to solve all the problems of cache coherence that shared memory hardware has, but now using the (relatively) slow messaging passing layer rather than custom-designed hardware.

The NUMA aspect is so unpredictable that many programmers prefer to be in control and have an explicitly non-shared model.
The underlying system also needs to solve all the problems of cache coherence that shared memory hardware has, but now using the (relatively) slow messaging passing layer rather than custom-designed hardware.

The NUMA aspect is so unpredictable that many programmers prefer to be in control and have an explicitly non-shared model.

When you write `FetchDouble` you *know* it is going to be slow.
The underlying system also needs to solve all the problems of cache coherence that shared memory hardware has, but now using the (relatively) slow messaging passing layer rather than custom-designed hardware.

The NUMA aspect is so unpredictable that many programmers prefer to be in control and have an explicitly non-shared model.

When you write `FetchDouble` you *know* it is going to be slow.

Compare with “how fast is $x = y$?” in VSM.
The underlying communications layer in VSM might be implemented
The underlying communications layer in VSM might be implemented

- in the Operating System, such as Mosix. This means all standard system libraries and user code can be used unchanged and a cluster looks like a single big machine
The underlying communications layer in VSM might be implemented

- in the Operating System, such as Mosix. This means all standard system libraries and user code can be used unchanged and a cluster looks like a single big machine
- by the programming language and libraries, such as Cluster OpenMP or Unified Parallel C (see later), so the language may need a bit of learning
VSM is currently fairly rare in practice, though as NUMA techniques improve, people are starting to talk about shared memory clusters as being a viable and useful way to proceed.
Latency numbers every programmer should know

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time (ns)</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 Cache hit</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>Mutex lock/unlock</td>
<td>25</td>
<td>25</td>
</tr>
<tr>
<td>Main memory access</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Read 1MB from memory</td>
<td>250,000</td>
<td>2.9 days</td>
</tr>
<tr>
<td>Round trip within datacentre</td>
<td>500,000</td>
<td>5.8 days</td>
</tr>
<tr>
<td>Read 1MB from disk</td>
<td>30,000,000</td>
<td>1 year</td>
</tr>
<tr>
<td>Send a packet California → California</td>
<td>150,000,000</td>
<td>4.8 years</td>
</tr>
</tbody>
</table>

Data source: [https://gist.github.com/hellerbarde/2843375](https://gist.github.com/hellerbarde/2843375)