NUMA in OpenMP - Home Sweet Home
Part I - What is NUMA?

Ruud van der Pas, Oracle Linux Engineering
My background is in mathematics and physics

Previously, I worked at the University of Utrecht, Convex Computer, SGI, and Sun Microsystems

Currently I work in the Oracle Linux Engineering organization

I have been involved with OpenMP since the introduction

I am passionate about performance and OpenMP in particular
What is NUMA?
NUMA stands for Non-Uniform Memory Access

Simply said, this means that the time it takes to fetch data from memory is not a constant

This time depends on where the data physically resides in memory

Let’s look at a typical NUMA system
A Generic, but Contemporary NUMA System

![Diagram of a NUMA system with nodes, cores, memory, and scalable bandwidth connections.](image-url)
Hardware Threads

Node

Cores

Memory

Core

Hardware Threads
NUMA - The Developer’s View
NUMA - Local versus Remote Access Times

My Thread Executes Here

Local Access (Fastest)

Remote Access (Slower)
Why NUMA Tuning Matters
Tuning for NUMA improves the performance by a factor of 22!
Home Sweet Home
In a NUMA system, every memory page has a home node.

The placement policy controls the location of the home node.

A commonly used policy is called “First Touch”.
The **First Touch** placement policy allocates the data page in the memory closest to the thread accessing this page for the first time.

This policy is the default on Linux and other OSes.

It is the right thing to do for a sequential application.

It is not always the right choice in a parallel application though.

The data may end up on a single node.
A first step towards tuning for NUMA is to avoid that all data ends up in the memory of a single node.

This can easily happen if the data initialization is sequential.

A single thread then touches the data and sets the home node.

Luckily, often there is an easy solution.
**Parallelize the data initialization part!**

```c
#pragma omp parallel for schedule(static)
for (int i=0; i<n; i++)
a[i] = 0;
```

Each thread has a slice of vector “a” in its local memory*

*) The allocation is on a virtual memory page basis
Thread affinity provides for a way to specify where threads should execute

Such controls are often used to get threads close to their data

On Linux, tools like “numactl” can be used for this

OpenMP provides its own set of affinity controls
Other Scenarios to Watch Out For

<table>
<thead>
<tr>
<th>In case data is read from file, a <strong>redundant parallel initialization</strong> can do magic, because this defines the home node(s) in advance</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>A malloc() call does not touch the data</strong></td>
</tr>
<tr>
<td><strong>Make sure that the thread that needs this block, initializes it</strong></td>
</tr>
<tr>
<td><strong>A calloc() call outside of a parallel region may cause the data to be on a single node</strong></td>
</tr>
</tbody>
</table>
Hopefully this talk has helped to clarify what NUMA is about

We also introduced some key concepts related to NUMA

In the second part of this talk, we show how OpenMP can be used to leverage a NUMA architecture
Thank You And ... Stay Tuned!

Bad OpenMP
Does Not Scale

Ruud van der Pas
SC’21 OpenMP Booth Talk
SC’21 Booth Talk Series

openmp.org  OpenMP API specs, forum, reference guides, and more

link.openmp.org/sc21  Videos and PDFs of OpenMP SC’21 presentations