Multithreading and multi-core CPUs (RV64IMFDQVA_Zicsr_Zifencei aka RV64GQV)

Adam Boult (www.bou.lt)

July 31, 2025

Contents

Preface

This is a live document, and is full of gaps, mistakes, typos etc.

Part I

Parallel programming

Chapter 1

Parallel processing

1.1 Parallel processing

1.1.1 MapReduce

Framework for parallising problems.

Say we have text and we want to do a word count. We can split the text into multiple subsections, do a word count on each and add these up.

MapReduce can do this.

1: Splitting

The main server splits the database into subsections.

For example, the text document could be divided into M chunks, to be split across different servers.

The master server can then pass data to be Mapped when they have capacity. Each server could run Map on a large number of documents.

Each document here is stored as a key value pair. The value here would be the text, and the key would identify the document.

$\{2 : "AppleAppleCarrot"\}$

2: Mapping

Rather than simply doing a word count on the subsections, we do a more complex, but more efficient approach.

We map the key value pair to a list of key value pairs. In our example we would map this to identifiers for each word.

The output here would be

$[\{"Apple" : 1\}, \{"Apple" : 1\}, \{"Carrot" : 1\}]$

The Map function will differ depending on the use case.

3: Shuffling

We now want to combine the key value pairs. There are R keys from the intermediate output of the Map program.

For each of the R keys, the master server assigns a server to reduce the key.

The master server then notifies the Reduce worker of the location of each relevant document output from the Map process. The Reduce server then copies the key value pair.

The Reduce worker now has an input of key value pairs. In our example one would have received:

$\{"Apple" : 1\}$

$\{"Apple" : 1\}$

It would also have received other instances from other servers.

The Reduce worker can then combine these to form the following:

$\{"Apple" : [1, 1]\}$

4: Reduce

Once all the data has been mapped, the data can be reduced.

In our example this involves creating:

$\{"Apple" : 2\}$

The Reduce function will differ depending on the use case.

5: Output

The outputs of each reduce can then be shared back to the main server.

We now have our word count.

Part II

Compute Unified Device Architecture (CUDA) and General-Purpose computing on Graphics Processing Units (GPGPU)

Part III

Tensor Processing Units (TPUs)

Part IV

SORT

Chapter 2

SORT 2025

2.1 Introduction

2.1.1 Introduction

big page, maybe in compiled or parallel specific stuff. cuda, opencl, cuBLAS, rocBLAS. actually probably want to separate stuff in base headered c and related, and GPU stuff. standard portable intermediate representation (SPIR) used in vulk and opencl ROCm Radeon Open Compute platforM oneAPI. open standard for GPUs, ai accelerators etc "and cuda" in big title of the gpu stuff opencl and sycl. same big page as cuda. + integrated graphics gpu memory nvidia: nvidia-smi program

Instruction level parallelism in that page. How much a program can be parallelised simulatneous multithreading as concept relaed to superscalar CPUs (multiple execution units)

c parallel

```
#include <pthread.h>
pthread\_t threads[2];

// start
pthread\_create(&threads[0], NULL, myFun, (void *) &thread\_args);
// block until thread finished
pthread\_join(threads[0], NULL);
```

in big page on parallel stuff, h3 on linear algebra on parallel: hyperthreading