

Notes for Santander1, the first lecture of Stephen Wolbers:

Overall:

The purpose of this lecture is to lay out some of the issues involved in designing and building computing systems for large computing problems in HEP. This includes many different types of problems and this lecture tries to survey some of the major ones – L3, reconstruction, simulation, secondary dataset generation and data analysis. Most of these are pretty straightforward to characterize, with the notable exception of physics analysis.

Slide 2:

The outline shows the goals of the talk. First there will be a general overview, then the talk will step through 5 different types of large scale computing in HEP and finally a short description of what has been built in some specific cases. Due to the fact that Heidi and I are both working at Fermilab (D0 and CDF respectively) the examples and emphasis will necessarily be on Fermilab and US experiment and computing.

Slide 3:

This shouldn't need too much explanation. Computing is important, we couldn't do HEP without it, and never have.

Slide 4:

A more general example of why computing is important to science. I found this in the New York Times back in March and found it was interesting. HEP has always used computing very heavily to do its science. It is becoming more and more important as time goes by that other fields of science (all fields of science?) are headed in the same direction. We (HEP) are merely some years ahead, but others are catching up and are more advanced in many cases.

Slide 5:

This slide is something I've been using for quite some years. There is no way HEP could have advanced and continues to advance if it wasn't for similar advances in computing. Clear examples can be found all over the place. Most experiments when they find they can record and then analyze more data than they expected will do so. That in turn leads to better capabilities (rarer signals can be studied, more quantities can be measured, etc.) which in turn leads to improved detectors which collect even more data.

Slide 6:

Maybe not necessary. Just a very simple picture of how the field works.

Slides 7&8:

I found these somewhere and liked them very much. It shows how the theory and the experiment are connected and what the detectors actually collect. On the bottom is the way that computing is used to make the connections between nature (theory) and detector output (numbers).

Slide 9:

This slide is meant to introduce the issues of distributed computing. In general there is some source of data (on the left), some computing (the middle), and some final results (on the right). The computing system is very sketchily represented by a group of boxes. The input is data and associated databases. The output includes datasets of many kinds and formats as well as associated logfiles, ntuples, histograms, database updates, etc etc.

Slide 10:

The data is complicated and it is important to know many things about the data in order to properly design and build a system that is meant to read that data. The questions here represent some of the things that are needed before one can properly design a system. In many cases somebody else determines the answer to these questions and the system meant to handle the files has to be built to accommodate these data characteristics.

Slide 11:

The computing system consists of many things, CPUs, memory, disks, tape storage, networks, etc. Designing this system is not simple and it is not unique. It is certainly possible to built systems that solve the computing problem with more than one architecture. Nevertheless, the answers to these questions are necessary and need to be used when a big system is designed.

Slide 12:

The output is important. The format, size, type, etc. are all important. The location of the data is important. The type of data is important. The movement and the access to that data is important. In many cases the output is the most difficult problem of all in building a system.

Slide 13-14:

This begins the examination of different types of computing. Level 3 (or any pure software trigger) is discussed here. In general these systems are CPU limited and the decisions that are made are not reversible. Ideally one would run the entire offline code with full calibrations and then make cuts based on physics quantities of interest. In the absence of enough CPU and final calibrations the decisions are based on less complete reconstruction.

Slide 15:

A diagram of the CDF L3 system. Data comes from the bottom of the figure. The data is moved over commodity networks (ATM and Ethernet) to the event-building system and L3 nodes (PC/LINUX) and the output is written to disk by the data logger. The data is later written to tape in the Feynman computing center. The ratio of input to output in CDF L3 is $300/75 = 4$.

Slide 16:

A slide showing the L1/L2/L3 triggering system (in rate) in CMS.

Slide 18:

Some numbers to characterize L3 for CDF and CMS.

Slide 20:

Monte Carlo is the second problem described. It is a large CPU problem and has a large CPU/data ratio. These make it an ideal computing problem to distribute all over the world and is in no way necessary to do it in a carefully designed central facility (though there is no reason it cannot be done there).

Slide 22-25:

There is a bit of a digression to discuss SpecInt, instructions/byte etc. It should be fairly clear. Most of HEP code is heavily integer (this is known empirically) and so a benchmark like SpecInt95 is a good one to characterize the expected performance of code on a computer. Given that, one can compute for a code a number – instructions/byte. This is defined to be $\text{SpecInt95} * 40 * 1,000,000 / \text{data size in bytes}$. This number is important and can be used to decide whether a specific code is I/O bound (low instructions/byte) or CPU bound (large instructions/byte). Over the

years it has been seen that instructions/byte was similar for most experiments, independent of the type of detector or event size.

Slide 26-28:

A few slides describing some details of Monte Carlo. Even though the simulation is easy to run almost anywhere ensuring that the correct code and parameters are being used is not necessarily simple.

Slide 29:

Event reconstruction is the next topic. The problem is well-defined and the systems that are used are all pretty much the same. This does not mean that it is a trivial problem and it does take a fairly large amount of work to design a system that can handle the large amount of data using the proper code, calibrations, etc. The output can be especially difficult to deal with, as the desired goal is to produce data which is useful for physics analysis, which is organized in an efficient manner for physics analysis, and is efficiently accessed.

Slides 30-32:

Event reconstruction involves very large amounts of data and CPU power. Most big experiments are well over 100 Tbytes/year and growing. This has been true for a while, as some of the fixed-target experiments at CERN and Fermilab have built high capacity DAQ systems and wrote large datasamples in past runs. The instructions/byte is large and possibly growing with time. It is not known whether this is an effect of having more CPU available, more complicated detectors, more difficult events (lots of overlays in LHC), C++ code, or some combination of those.

Slides 33-34:

A short summary of event reconstruction and a reminder that lecture 2 will describe design and construction of farms for reconstruction.

Slides 35-38:

The production of secondary datasets is usually done to provide compressed and enriched datasets. The CPU to I/O ratio is normally quite low here and systems that are used to make this selection and write those new sets have to be designed to take this into account.

Slide 39:

Physics analysis is notoriously difficult to characterize and to build systems that satisfy all the large demands placed upon it. Compared to the previous 4 types of systems it has the largest user community by far and that community is constantly trying to get the most out of the system – the demand is not predictable and it is not constant. As an experiment takes data and matures the patterns of access and use are constantly changing.

Slide 40-41:

This slide describes how physics analysis used to be done, slide 41 represents in a very sketchy way what a physics analysis system looks like. Essentially all components are connected via the network to the resources that are required. The challenge is to build such a system and have it provide the capabilities required. GRID computing certainly can be represented as part of this model.

Slides 42-54:

Theoretical descriptions of big systems are very interesting and useful. However, at some point real hardware has to be purchased and real software systems need to be written and integrated into these systems, experiment code must be written and used, data flows established, databases installed, etc etc. These slides look at just a few systems for some of the big experiments taking data now, including CDF and D0 at Fermilab, RHIC, and BABAR. Some of the pictures are a little out-of-date but they do give an indication of what is done in reality when people are faced with a deadline of producing working systems consistent with money and people available to do the work.

Slide 55:

A presummary.

Slide 56:

The summary.