Formula Based Performance Design

-

Gerrit Muller

HSN-NISE

Hasbergsvei 36 P.O. Box 235, NO-3603 Kongsberg Norway

gaudisite@gmail.com

Abstract

Performance models are mostly simple mathematical formulas. The challenge is to model the performance at an appropriate level. In this presentation we introduce several levels of modeling, labeled zeroth order, second order, et cetera. AS illiustration we use the performance of MRI reconstruction.

All Gaudí documents are available at:
http://www.gaudisite.nl/

1 Introduction

We recommend to model performance by using simple, secondary school, mathematical formulas. Frequently designers tend to start using more advanced techniques and formalisms, in an attempt to be accurate. In this paper we discuss an approach using simple mathematical formulas, starting with the most simple formulas and refining these formulas as far as needed.

2 Using n-order formulas

The basis for most performance models are simple mathematical formulas, using secondary school math. The challenge is to keep the models as simple as possible, as discussed in the section about control design. We can express the degree of detail in formulas by the order of the formula. Figure 1 shows such classification.
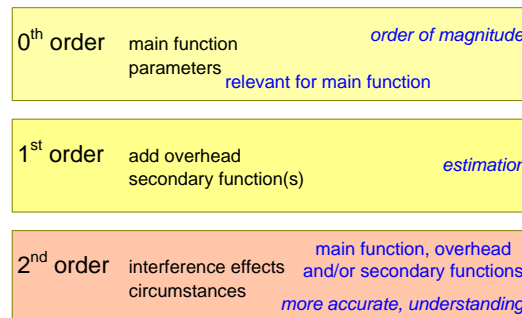

Figure 1: Theory Block 1: n Order Formulas

Figure 2 shows an example of a highly simplified model of the CPU load for image processing. This formula assumes that the CPU load is directly proportional to the number of pixels plus some time to perform the user interface tasks. We call such a formula, where only the main parameter is present, a zeroth order formula.

$$t_{cpu\ total} = t_{cpu\ processing} + t_{UI}$$

$$t_{cpu\ processing} = n_x * n_y * t_{pixel}$$

Figure 2: CPU Time Formula Zero Order

It could be that the 0-order formula does not work well enough, for example because overhead is significant. In Figure 3 the biggest overhead contribution is added to the formula, in this example the context switch overhead.

$$t_{\text{cpu total}} \quad = \quad t_{\text{cpu processing}} \quad + \quad t_{\text{UI}}$$

$$+ \quad t_{\text{context switch overhead}}$$

Figure 3: CPU Time Formula First Order

However, in a heavily loaded system may suffer additional loads due to the context switches, the so-called second order effects. In Figure 4 these second order effects are added to the formula. The second order impact may depend on the type of system load. The second order terms might be parameterized to express this relation. For example signal processing loads might cause low penalties, due to high cache efficiency, while control processing might be much more sensitive to these effects.

$$t_{\text{cpu total}} \quad = \quad t_{\text{cpu processing}} \quad + \quad t_{\text{UI}} \quad + \quad t_{\text{context switch overhead}} \quad +$$

$$t_{\text{stall time due to cache efficiency}} \quad + \quad t_{\text{stall time due to context switching}}$$

signal processing: high efficiency
control processing: low/medium efficiency

Figure 4: CPU Time Formula Second Order

3   Example of n-order formulas in MR reconstruction

The reconstruction of MR images is a processing intensive operation. Fast reconstructions are beneficial for the throughput of MRI scanners and are prerequisite for a number of performance critical applications. Figure 5 shows a simplified block diagram of an MRI scanner, the context of the MR reconstruction. The MR data is digitized in the acquisition subsystem and transferred to the reconstruction subsystem. The reconstructed images are stored in the data base and viewed at the operator or viewing console. All subsystems are controlled by a central host computer.

In Figure 6 a visualization and mathematical formulas are used in combination to model the performance of the MR reconstruction. The visualization shows the
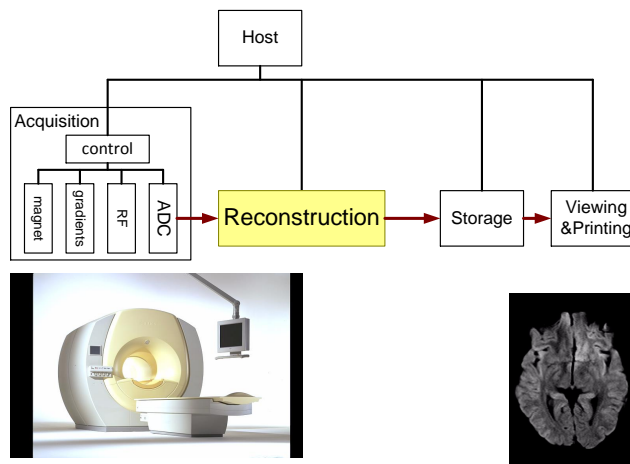
Figure 5: MR Reconstruction Context

$$t_{recon} = n_{raw\text{-}x} * t_{fft}(n_{raw\text{-}y}) + n_y * t_{fft}(n_{raw\text{-}x})$$

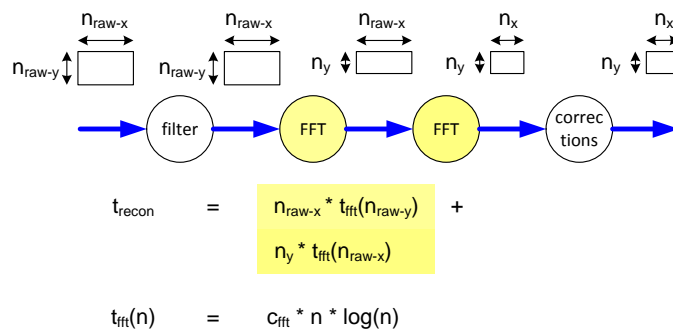$$t_{fft}(n) = c_{fft} * n * log(n)$$

Figure 6: MR Reconstruction Performance Zero Order

processing steps that are performed as reconstruction. Above the arrows it is shown what the size of the data matrices is at that phase.

This 0-order model uses the Fast Fourier Transform (FFT) as the dominating term contributing to the performance Most operations are directly proportional to the matrix size shown above the formulas. The FFT itself is an order $nlog(n)$ term, parameterized with its corresponding load c_{fft}.

Unfortunately the formulas don't tell us much without quantification. Figure 7 provides us with some quantified input based on a FFT micro-benchmark: an FFT on thousand points executes in about 5 msecs (typical performance figures for processing hardware around 1990). The figure takes one typical use case, where a 512*256 raw image is reconstructed on a 256*256 image, to calculate the reconstruction performance. For this use case and assumptions we get 1.2 seconds.

Typical FFT, 1k points ~ 5 msec
(scales with 2 * n * log (n))

using:

$n_{raw\text{-}x}$ = 512

$n_{raw\text{-}y}$ = 256

n_x = 256

n_y = 256

t_{recon} = $n_{raw\text{-}x}$ * $t_{fft}(n_{raw\text{-}y})$ +

n_y * $t_{fft}(n_{raw\text{-}x})$ +

512 * 1.2 + 256 * 2.4

~= 1.2 s

Figure 7: Zero Order Quantitative Example

$n_{raw\text{-}x}$ $n_{raw\text{-}x}$ $n_{raw\text{-}x}$ n_x n_x

$n_{raw\text{-}y}$ $n_{raw\text{-}y}$ n_y n_y n_y

filter → FFT → FFT → corrections

t_{recon} = $t_{filter}(n_{raw\text{-}x}, n_{raw\text{-}y})$ +

$n_{raw\text{-}x}$ * $t_{fft}(n_{raw\text{-}y})$ +

n_y * $t_{fft}(n_{raw\text{-}x})$ +

$t_{corrections}(n_x, n_y)$

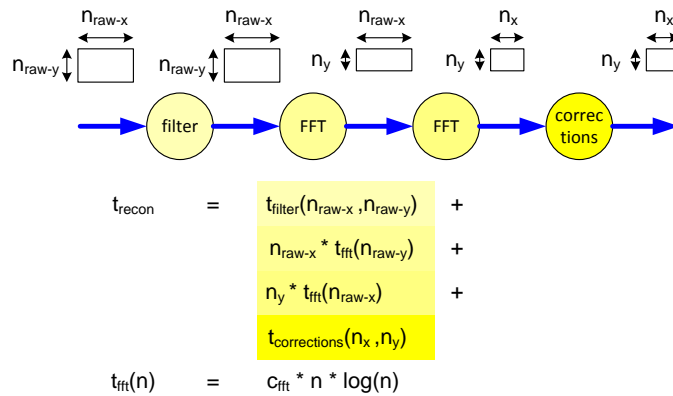$t_{fft}(n)$ = c_{fft} * n * $log(n)$

Figure 8: MR Reconstruction Performance First Order

Figure 9 extends the model to also take the non-FFT processing into account. These operations filter the raw data and perform some simple corrections on the image. Both operations are proportional to the number of pixels that is processed.

Figure 9 provides the quantifications obtained by micro-benchmarking both operations: 2 msec to process 1k points. Using the same numbers as Figure 7 we get for filtering $512 * 256 * 2/1024ms \approx 0.26s$ and for correction $256 * 256 * 2/1024ms \approx 0.13s$. Both processing steps can not be ignored compared to the FFT operation!

Finally we add bookkeeping and I/O type operations to the formula, see Figure 10. In practice both terms often ruin the performance of well designed processing kernels, mostly by a lack of attention.

Typical FFT, 1k points ~ 5 msec
(scales with 2 * n * log (n))

Filter 1k points ~ 2 msec
(scales linearly with n)

Correction ~ 2 msec
(scales linearly with n)

Figure 9: First Order Quantitative Example

$$t_{recon} = t_{filter}(n_{raw-x}, n_{raw-y}) + n_{raw-x} * (t_{fft}(n_{raw-y}) + t_{col-overhead}) + \quad\quad t_{fft}(n) = c_{fft} * n * \log(n)$$

$$n_y * (t_{fft}(n_{raw-x}) + t_{row-overhead}) + t_{corrections}(n_x, n_y) + t_{read\ I/O} + t_{transpose} + t_{write\ I/O} + t_{control-overhead}$$

focus on overhead reduction

is more important

than faster algorithms
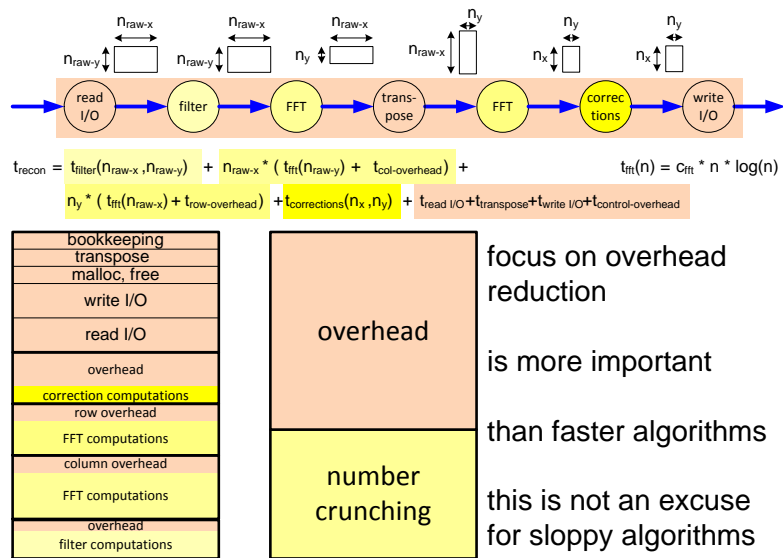
this is not an excuse for sloppy algorithms

Figure 10: MR Reconstruction Performance Second Order

4   Summary

We have shown that performance can be modeled by starting with the formula for the main function and its parameters. This formula is refined by adding factors that significantly contribute to the (non-)performance.

We used MRI reconstruction as an example of these types of formulas. The formulas are limited to multiplications and logarithms. In this example we have to add quite some factors outside of the main functionality to obtain a usable performance model: simple correcion and filter functions, bookkeeping, data-restructuring, and input/output. We also showed that the formulas provide insight, especially the impact of the different parameters, but that actual quantifications also add insight in actual performance numbers and the relevance of the different terms.

5 Acknowledgements

The diagrams are a joined effort of Roland Mathijssen, Teun Hendriks and Gerrit Muller. The approach is based on the EXARCH course created by Ton Kostelijk and Gerrit Muller.

References

[1] Gerrit Muller. The system architecture homepage. `http://www.gaudisite.nl/index.html`, 1999.

History
Version: 1.0, date: 3 September, 2007 changed by: Gerrit Muller
- Created text by copying from PerformanceEngineering
- changed logo to PHRTreconstructionMRzeroOrder
- changed status to draft

Version: 0, date: 17 February, 2007 changed by: Gerrit Muller
- Created, via refeactoring of ASP course, no changelog yet