Formula Based Performance Design

by *Gerrit Muller*   HSN-NISE
e-mail: gaudisite@gmail.com
www.gaudisite.nl

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.

August 21, 2020
status: draft
version: 1.0

Theory Block: n Order Formulas

0^{th} order — main function parameters

order of magnitude

relevant for main function

1^{st} order — add overhead secondary function(s)

estimation

2^{nd} order — interference effects circumstances

main function, overhead and/or secondary functions

more accurate, understanding

CPU Time Formula Zero Order

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

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

CPU Time Formula First Order

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

$$+ \; \boxed{t_{\text{context switch overhead}}}$$

CPU Time Formula Second Order

$$t_{\text{cpu total}} = t_{\text{cpu processing}} + t_{\text{UI}} + t_{\text{context switch overhead}} + \boxed{t_{\text{stall time due to cache efficiency}} + t_{\text{stall time due to context switching}}}$$

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

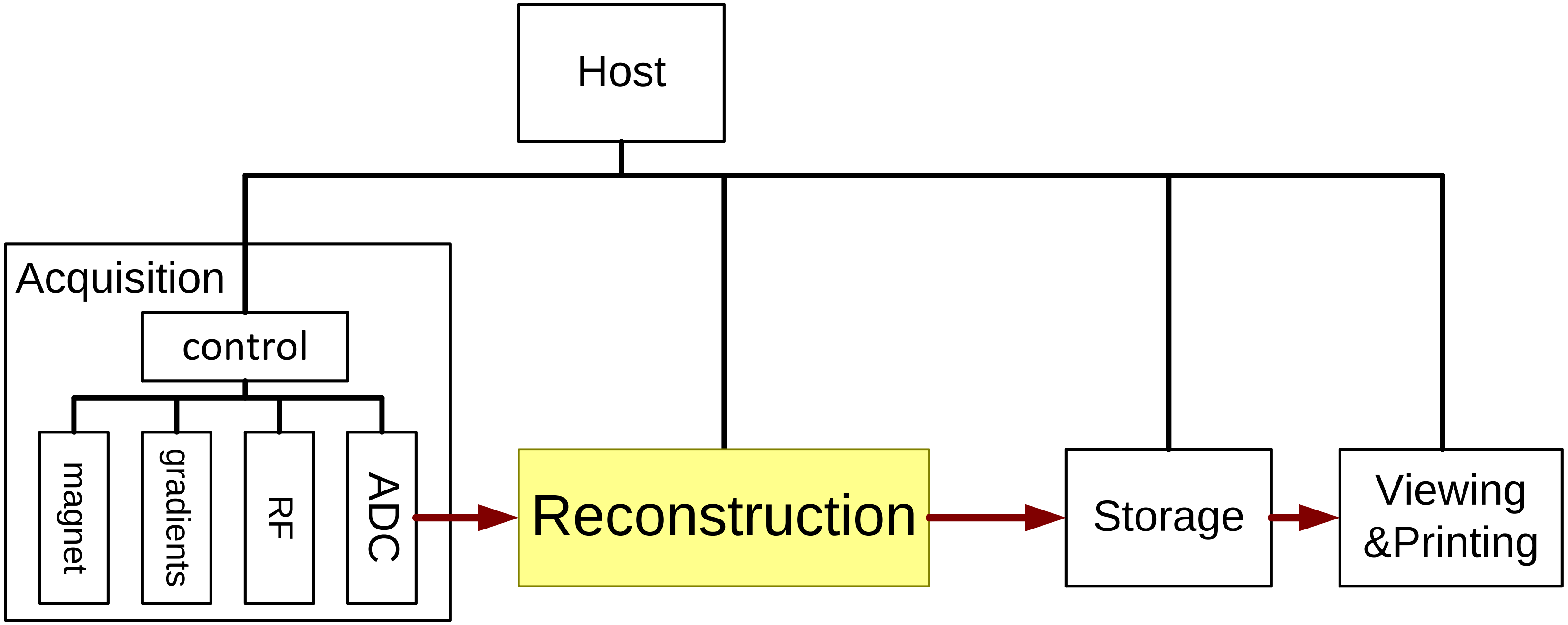
Case MRI Reconstruction

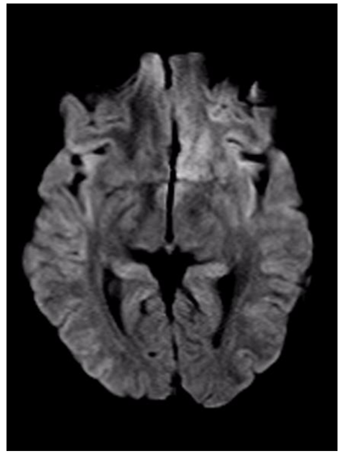
MRI reconstruction

"Test" of performance model on another case

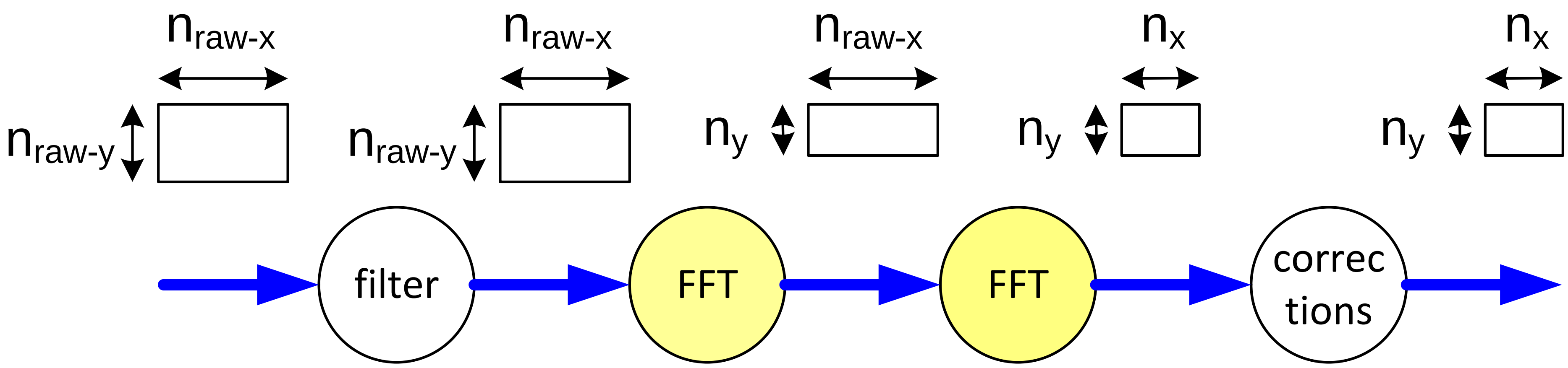
Scope of performance and significance of impact

MR Reconstruction Context





MR Reconstruction Performance Zero Order



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

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

Typical FFT, 1k points ~ 5 msec

(scales with 2 * n * log (n))

using:

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

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

$n_x = 256$

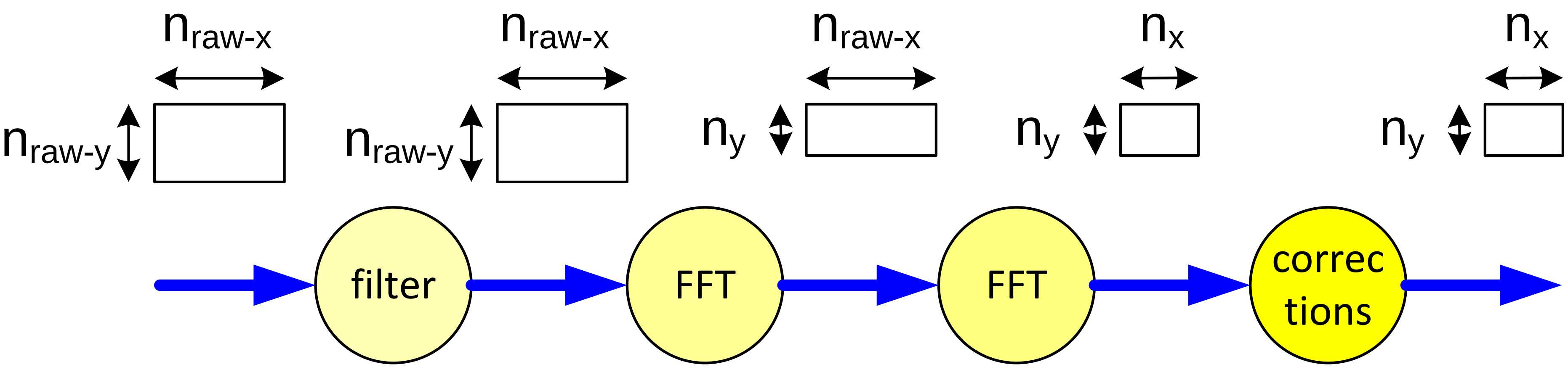
$n_y = 256$

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

512 * 1.2 + 256 * 2.4

~= 1.2 s

MR Reconstruction Performance First Order



$$
\begin{aligned}
t_{recon} \quad = \quad & t_{filter}(n_{raw-x}, n_{raw-y}) \quad + \\
& n_{raw-x} * t_{fft}(n_{raw-y}) \quad + \\
& n_y * t_{fft}(n_{raw-x}) \quad + \\
& t_{corrections}(n_x, n_y)
\end{aligned}
$$

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

Typical FFT, 1k points ~ 5 msec

(scales with 2 * n * log (n))

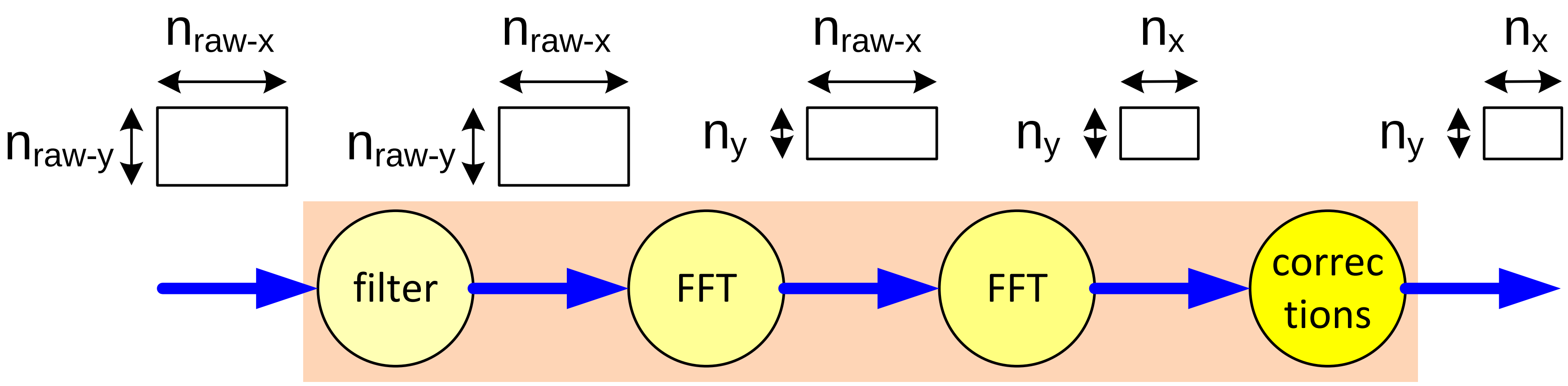
Filter 1k points ~ 2 msec

(scales linearly with n)

Correction ~ 2 msec

(scales linearly with n)

MR Reconstruction Performance Second Order



$$
\begin{aligned}
t_{recon} = \; & t_{filter}(n_{raw-x}, n_{raw-y}) \; + \\
& n_{raw-x} * (\, t_{fft}(n_{raw-y}) \; + \; t_{col-overhead} \,) \; + \\
& n_y * (\, t_{fft}(n_{raw-x}) \; + \; t_{row-overhead} \,) \; + \\
& t_{corrections}(n_x, n_y) \; + \\
& t_{control-overhead}
\end{aligned}
$$

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

Second Order Quantitative Example

Typical FFT, 1k points ~ 5 msec

(scales with 2 * n * log (n))

Filter 1k points ~ 2 msec

(scales linearly with n)

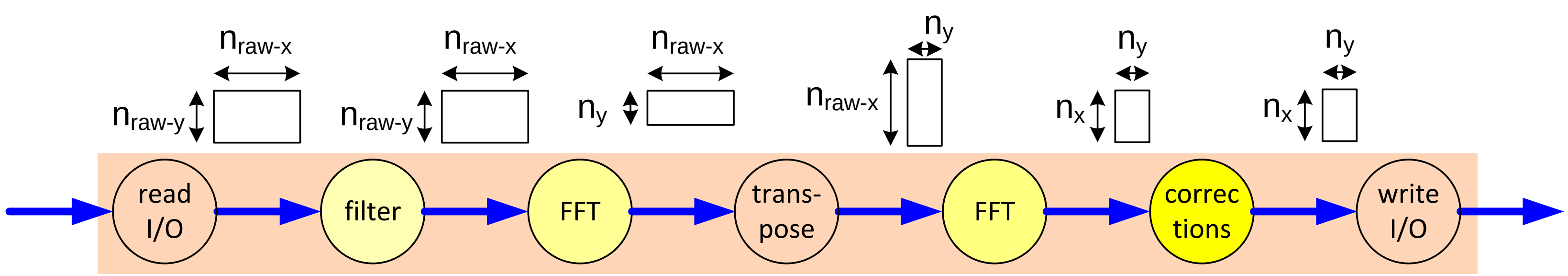
Correction ~ 2 msec

(scales linearly with n)

Control overhead = $n_y * t_{row\ overhead}$

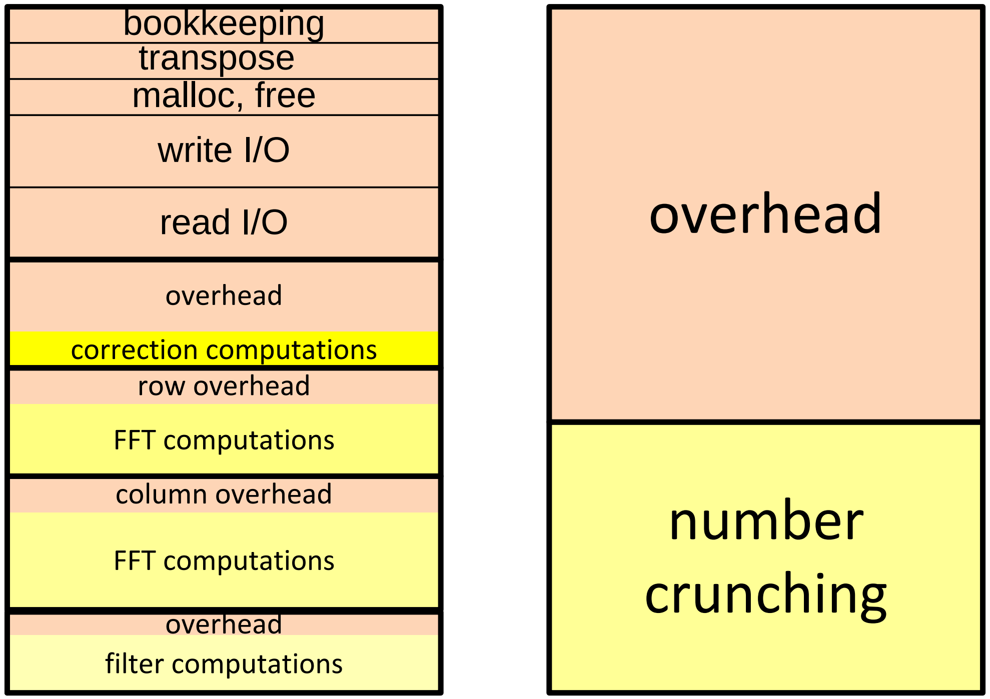
10 .. 100 µs

MR Reconstruction Performance Third Order



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

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



focus on overhead reduction

is more important

than faster algorithms

this is not an excuse for sloppy algorithms

Summary Case MRI Reconstruction

MRI reconstruction

System performance may be determined by other than standard facts

E.g. more by overhead I/O rather than optimized core processing

==> Identify & measure what is performance-critical in application

Colofon

The ASP™ course is partially derived from the EXARCH course developed at *Philips CTT* by *Ton Kostelijk* and *Gerrit Muller*.

Extensions and additional slides have been developed at *ESI* by *Teun Hendriks*, *Roland Mathijssen* and *Gerrit Muller*.