Time-Cost Trade-offs of Pipelined Dataflow Applications

Jonathan Kho[1], **Erik Saule**[4], Anas AbuDoleh[1], Xusheng Wang[1], Hao Ding[2], Kun Huang[1], Raghu Machiraju[1,2], Ümit V. Çatalyürek[1,3]

[1] Biomedical Informatics, The Ohio State University
[2] Computer Science and Engineering, The Ohio State University
[3] Electrical and Compute Engineering, The Ohio State University
[4] Computer Science, UNC Charlotte

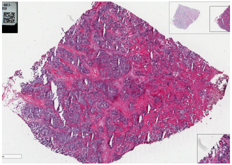
Nashville 2016

Context

- Cloud computing promises on demand resources
- Different types of computing resources are available
- Arbitrary speedups are in principle possible
- The catch is that you have to pay for resources used
- The problem becomes a **tradeoff** between the **runtime** of an application and **cost** of executing it

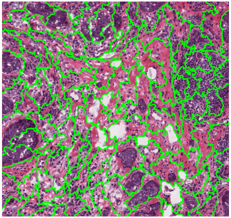
In this presentation, we show that the **pipelined dataflow abstraction** is good for expressing this tradeoff because runtime is “easy” to predict. We use a particular imaging application to examplify the technique.

Feature Extraction from Histopathological Slides



Biopsy slides

Non-blank patch Preprocess



SuperPixel segmentation

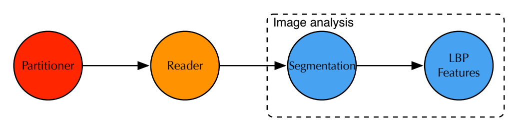
LBP feature extraction

- Varying sizes in the order of 100k $\times$ 100k pixels.
- Aperio Format with thumbnail (about 1GB/file, 24GB uncompressed)
- Available public repository (TCGA) with 1000s of participants samples
  - 3 slides per patients.
- Can be used to predict whether the biopsy is cancerous
- Will consider two instances: twoparticipants (2 participants) and allslides (42 participants)

Outline

Pipelined workflow

Layout

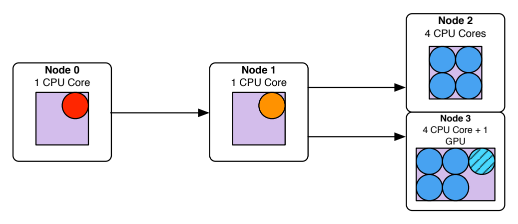


Reader discards background tiles

Advantages

- Sequential processes
- Heterogeneous
- Replication for throughput
- Comm/Comp overlap

Placement



Application

- Medical imaging
- Stock option pricing
- Synthetic Aperture Radar
- Incremental graph algorithm

How to predict runtime ?

In a pipelined system what matters is the steady state! The throughput is given by the most loaded node.

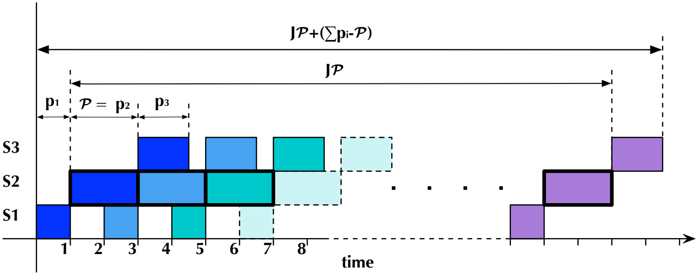
Runtime in a (simple) pipelined dataflow model

Model

- An application of M stages
- J identical jobs
- Stage i processes a job in p_i

One-to-one mapping

- With one processor per stage
- The execution is constrained by the slowest stage
- Period $\mathcal{P} = \max_i p_i$
- Throughput $T = \frac{1}{\mathcal{P}}$



Runtime in a (more complex) pipelined dataflow model

Replication

It is possible in some application to replicate some stages to increase the throughput

- If stage i is replicated r_i times
- i processes at a rate $\tau_i = \frac{r_i}{p_i}$
- Throughput $T = max_i \tau_i$
- Period $\mathcal{P} = \frac{1}{T}$

Runtime in a (more complex) pipelined dataflow model

Replication

It is possible in some application to replicate some stages to increase the throughput

- If stage i is replicated r_i times
- i processes at a rate $\tau_i = \frac{r_i}{p_i}$
- Throughput $T = max_i \tau_i$
- Period $\mathcal{P} = \frac{1}{T}$

Heterogeneity

It is possible in some application to replicate on different systems.

- If stage i is replicated on a CPU and a GPU
- i processes at a rate $\tau_i = \frac{1}{p_i^{cpu}} + \frac{1}{p_i^{gpu}}$
- Throughput $T = max_i \tau_i$
- Period $\mathcal{P} = \frac{1}{T}$

Runtime in a (more complex) pipelined dataflow model

Replication

It is possible in some application to replicate some stages to increase the throughput

- If stage i is replicated r_i times
- i processes at a rate $\tau_i = \frac{r_i}{p_i}$
- Throughput $T = max_i \tau_i$
- Period $\mathcal{P} = \frac{1}{T}$

Heterogeneity

It is possible in some application to replicate on different systems.

- If stage i is replicated on a CPU and a GPU
- i processes at a rate $\tau_i = \frac{1}{p_i^{cpu}} + \frac{1}{p_i^{gpu}}$
- Throughput $T = max_i \tau_i$
- Period $\mathcal{P} = \frac{1}{T}$

These two techniques combine !

Experimental settings and model calibration

Machine

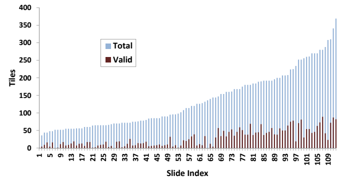
- 32-node cluster
- Two Xeon E5520 (quad core)
- An NVIDIA C2050
- DDR4x Infiniband

Software

- g++ 4.8.1
- mvapich2 2.2
- DataCutter (dcmpi)
- Openslide 3.4.1
- gSLIC
- nvcc 7.0.27

Tile prediction

based on thumbnail:



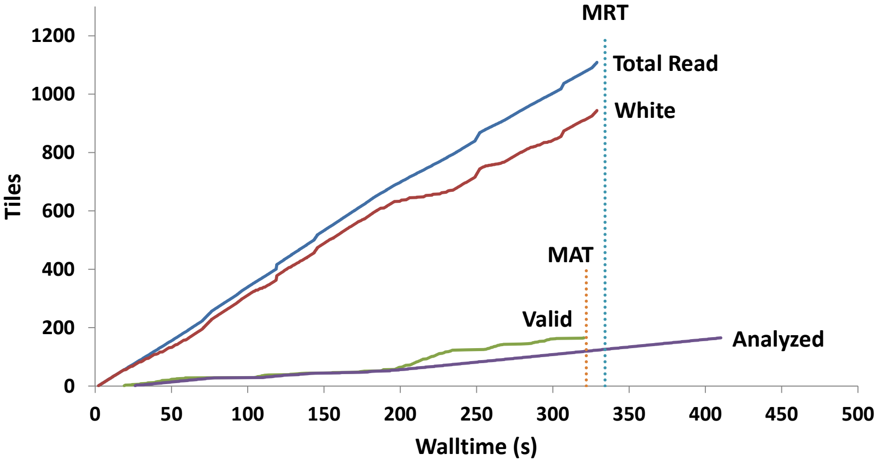
Model Calibration

Slide	Filesize	Width	Height	Estimated Total Tiles	Estimated Valid Tiles
TCGA-BH-A18V-01A-01-TSA	432.93MB	98,631	33,244	225	78
TCGA-BH-A18J-01A-01-TSA	322.01MB	112,037	29,845	224	75

ImAn:

CPU / GPU	Proc. Time	Local τ_{IA}	Average τ_{IA}	Speedup
NVIDIA Tesla C2050	447.41 s	2.924M px/s		
	422.03 s	2.981M px/s	2.953M px/s	1
Intel Xeon E5520 (7 cores)	399.11 s	3.278M px/s		
	378.83 s	3.321M px/s	3.299M px/s	1.117

A first log

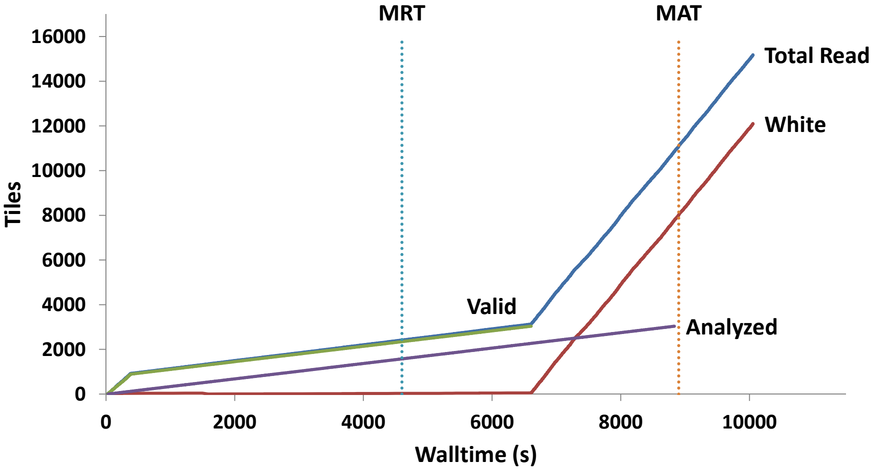


1 Reader. 3 GPUs. Two Patients. Natural ordering.
(Eventually ImAn idles because too many White are read.)

How to fix this ?

The Valid tiles are more computationally expensive than the White ones.
Valid first should work fine!

Valid First does not always work



1 Reader. 2 GPUs. All Slides. Valid First.

(The system has bounded memory and eventually Reader stalls.)

Outline

Flowshop

Deciding which job to process next in its simplest form is a Flowshop problem.

Model

- M stages
- J jobs
- job j in stage i takes time $p_{i,j}$
- Order the job to minimize the makespan

Bad News

- NP-Complete in this form
- That is actually an abstraction of the real problem

How to make the problem computationally simpler?

Since you have categories of jobs, the $p_{i,j}$ matrix is actually low rank.
That helped in $R||Cmax$. Maybe it helps here?

Interleave schedule

Insight

We have:

- C categories of jobs
- J_c jobs in category c
- J_c are large numbers

Sounds like something cyclic should work

Algorithm

Build k batches
with $s_c = \frac{J_c}{k}$ jobs of category c

Asymptotic optimality

Each batch can be seen as a meta job in a one-to-one mapping.
When k goes to infinity, the makespan of the flowshop problem converges to the optimal value of the pipelined scheduling problem.
So with lots of jobs, performance is good.

Dismissed Constraints

Divisibility

The number of jobs might be prime, but rational approximation works just fine.

Heterogeneity

Called hybrid problem in the flowshop world.
Heterogeneous just makes different $p_{i,j}$.

Onlineness

Non-clairvoyance can be solved with random ordering.

Low-Rank

Categories and low-rank are slightly different. (low rank admits linear combination of categories.)
Low-rank can be solved by some weighted interleave schedule

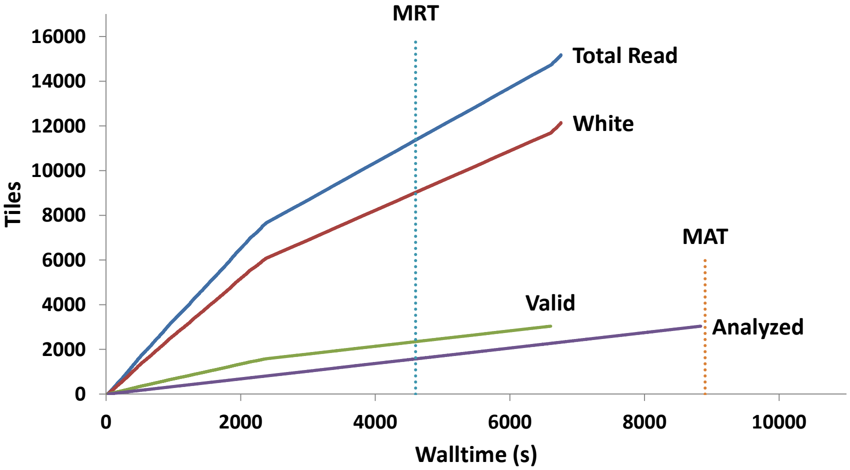
Communication

Often modeled as an additional stage of processing.

Blocking Writes

As long as one batch does not saturate memory, pipelining will happen gracefully.

In practice



1 reader. 2 GPU. all slide. interleave.

(All cases work just fine.)

Outline

Model

Time

Parameters:

- W white tiles
- V valid tiles
- r CPU Reader at a rate τ_{Read}^{CPU}
- c CPU ImAn at rate τ_{ImAn}^{CPU}
- g GPU ImAn at rate τ_{ImAn}^{GPU}

Prediction:

- $\tau_{Read} = r\tau_{Read}^{CPU}$
- $\tau_{ImAn} = c\tau_{ImAn}^{CPU} + g\tau_{ImAn}^{GPU}$
- $C_{max} = \max(\frac{W+V}{\tau_{Read}}, \frac{V}{\tau_{ImAn}})$

Cost

Amazon EC2 charges per hour
(MS Azure charges per minute)
So the charge is
$\lceil \frac{C_{max}}{3600} \rceil ((r + c) * C_c + g * C_g)$

What you can get

In EC2, you can get a cg1 instance with 2 NVIDIA M2050 and 8 Xeon core for \$2.1 per hour.
For the reader, you can use a c1.medium that gives a Xeon core for \$0.13 per hour.

$(1+\epsilon)$-approximation of Time-Cost

Cost under time constraint

If you set a cap T on time, then you obtain bounds

- $\tau_{Read} \geq \frac{W+V}{C_{max}}$
- and $\tau_{ImAn} \geq \frac{V}{C_{max}}$

So:

- $r\tau^{c1}_{Read} \geq \frac{W+V}{C_{max}}$
- $r \geq \frac{W+V}{\tau^{c1}_{Read} C_{max}}$
- and $g\tau^{cg1}_{ImAn} \geq \frac{V}{C_{max}}$
- $g \geq \frac{V}{C_{max}\tau^{cg1}_{ImAn}}$

Min cost: pick smallest r and g.

$(1+\epsilon)$-approximation of Time-Cost

Cost under time constraint

If you set a cap T on time, then you obtain bounds

- $\tau_{Read} \geq \frac{W+V}{C_{max}}$
- and $\tau_{ImAn} \geq \frac{V}{C_{max}}$

So:

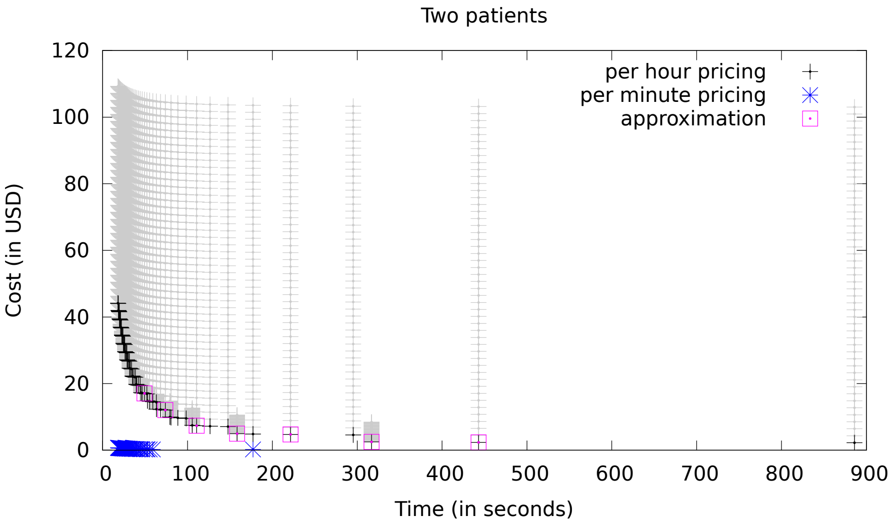
- $r\tau^{c1}_{Read} \geq \frac{W+V}{C_{max}}$
- $r \geq \frac{W+V}{\tau^{c1}_{Read} C_{max}}$
- and $g\tau^{cg1}_{ImAn} \geq \frac{V}{C_{max}}$
- $g \geq \frac{V}{C_{max}\tau^{cg1}_{ImAn}}$

Min cost: pick smallest r and g.

Pareto approximation

Using Papadimitriou and Yannakakis scheme.
Pick T_{min} and T_{max} and a basis $1+\epsilon$.
Return solution for $T = (1+\epsilon)^k T_{min}$ for all $k \in \mathbb{N}; 0 \leq k \leq \left\lceil \log_{1+\epsilon} \frac{T_{max}}{T_{min}} \right\rceil$
That set is a $(1+\epsilon)$ approximation of the Pareto set.

Some Values


Two patients
120
100
80
60
40
20
0
Cost (in USD)
0
100
200
300
400
500
600
700
800
900
Time (in seconds)
per hour pricing
per minute pricing
approximation

Outline

Conclusion

Predicting the runtime of pipelined dataflow application is feasible

- Simple bottleneck analysis should work
- Just make sure there are no artificial bubbles in the execution
- Integrates heterogeneous processors gracefully

Time Cost tradeoff in the cloud

- Once you have a closed formula for the runtime, picking the cheapest machine to finish the application in a given time is easy
- Finding an approximation of the Pareto-Curve is immediate

Future Works

Does low-rank matrices make flowshop easier ?

Here it works becasue we have lots of jobs.
Even in the hybrid case ?

Dynamic pricing

Spot instances have varying price in time.
Can we do a similar analysis with dynamic pricing ?

Power and Energy

There are works in pipelined execution with energetic objective.
Can we leverage them in practice ?

Thank you

More information

Contact : esaule@uncc.edu

Visit: `http://webpages.uncc.edu/~esaule`