Optimization Formalization

Sara Achour

May 29, 2014

1 Model V2

1.1 Hardware Model

Worker Machine $m \in M, |M|$
Main machine R

Machine cost for task with runtime t:

$c = a \cdot W \cdot t + b \cdot E \cdot t$
E: energy per unit time.
W: amount of work per unit time.
a: weight to apply to machine efficiency.
b: weight to apply to machine power consumption.

p_{fail} : probability of a crash per unit time.
p_{hang} : probability of a hang per unit time.
p_{err} : probability of soft data error occurring per unit time.
$p_{ok} = 1 - p_{fail} + p_{hang} + p_{err}$

Oversimplification: Assuming the probability of various failures is not dependent on the computation running.

1.2 Programming Model

taskset T:
size: |T|
number inputs: n
number outputs: m

task $t \in T$:
$t : R^m \rightarrow R^n$
task t is over real numbers. maps m dimension vector to n dimension vector.
$t : f(i_1, \cdot, i_n) \rightarrow (r_1, \cdot, r_m)$
i_k : kth input, let $\vec{i}$ be input vector.

r_k : kth output, let $\vec{r}$ be output vector.

$P(\vec{r}, \vec{i})$

The probability distribution of the outputs and inputs.

$t : f(i_1, \cdot, i_n) \sim argmax_{\hat{r}} P(\hat{r}|\hat{i})$

The input-output relation f can be rewritten as the expectation maximization of r given i.

Since f is deterministic, there should be a particular r where P(r,i) = 1 for every i.

ex: $P(\vec{r}|\vec{i})$

The probability of outputs r occurring, given input i.

δ_t: the runtime of the particular task.

1.3 Hardware-Programming Model Interaction

For a particular task t running on machine m:

$p^t_{fail} = \delta \cdot p_{fail}$

$p^t_{hang} = \delta \cdot p_{hang}$

$p^t_{err} = \delta \cdot p_{err}$

$P_{err}(\vec{r}, \vec{i})$

This is the output distribution of the task if an error occurs.

$P'(\vec{r}, \vec{i}) = P(\vec{r}, \vec{i}) + P_{err}(\vec{r}, \vec{i})$

the output distribution on machine m (P') is a mixture of the output distribution (P) distribution and the error distribution(P_{err}).

$P_\epsilon(\epsilon)$

this is the distribution of errors between the correct, faulty result. Note: we may want to optimize out this distribution, but for simplicity, I will make this directly accessible.

Question: given result $\vec{r} \sim P'(\vec{r}, \vec{i})$ and known $\vec{i}$ is it drawn from $P(\vec{r}, \vec{i})$ or $P_{err}(\vec{r}, veci)$?

if:

$P(\vec{r}, \vec{i}) > P_{err}(\vec{r}, \vec{i}) \rightarrow P$

otherwise

$\rightarrow P_{err}$

Now suppose we do not have $\vec{i}$. We find which distribution has the highest probability of producing $\vec{r}$ across all inputs.

let $P(\vec{r}) = \int_{\vec{i}} P(\vec{r}, \vec{i}) P(\vec{i})$

let $P_{err}(\vec{r}) = \int_{\vec{i}} P_{err}(\vec{r}, \vec{i}) P(\vec{i})$

if

$P(\vec{r}) > P_{err}(\vec{r}) \rightarrow P(\vec{r}, \vec{i})$

otherwise

$\rightarrow P_{err}$

Aside:

We don't have the actual probability distributions: P, P_{err}, we have incomplete approximations $\hat{P}, \hat{P}_{err}$ drawn from a series of observations $(r_{corr}, r_{err}) \in O$.

* We can pay to figure out which distribution the result comes from and improve estimates $\hat{P}$, $\hat{P}_{err}$.

1.4 Cost Function

Accuracy Requirement
$P(|f(\vec{i}) - f'(\vec{i})| < \epsilon) > \omega$
    ϵ: tolerated error
    ω : tolerated frequency of intolerable errors.

Relevant machine parameters:
    $P(|f(\vec{i}) - f'(\vec{i})| < \epsilon) \approx p_{err} \int_0^{\epsilon} P_\epsilon(\epsilon) d\epsilon$
    Compute the probability of an error occuring times the probability the error is less than epislon.

Semantics for cost function
    To differentiate between machines $m \in M$: $\{m_1, \cdots, m_n\} = M$
        $m_i \to \{P^i_\epsilon, p^i_{fail}, p^i_{hang}, p^i_{err}, c^i\}$
    To differentiate between tasks $t \in T$: $\{t_1, \cdots, t_n\} = T$
        There are no particular parameters to annotate, we assume we're talking about one taskset.
    The reliable machine has the following properties:
        $r \to \{P^r_\epsilon = 0, p^r_{fail} = 0, p^r_{hang} = 0, p^r_{err} = 0\}$

Accuracy Requirement for a task-machine mapping
  Given we have assignments, $t_i \in T | t_i \to m_j$:
    $P(|f(\vec{i}) - f'(\vec{i})| < \epsilon) = \frac{1}{|T|} \sum_{t_i = t_0}^{t_n} P_{m_j}(|f(\vec{i}) - f'(\vec{i})| < \epsilon)$
        The probability of a set of task-machine assignments is the average of each probability of each assignment meeting the assignment.

Minimization formula
    For a set of assignments $t_i \in T | t_i \to m_j$, we want to maximize:
        $\sum_{t_i \in T} c_j$

Putting it together:
    We want to find a set of $t_i \to m_j$ mappings that minimizes:
        $\sum_{t_i \in T} c_j$
    but meets the following constraint:
        $[\frac{1}{|T|} \sum_{t_i = t_0}^{t_n} P_{m_j}(|f(\vec{i}) - f'(\vec{i})| < \epsilon)] > \omega$
        ie:
        $[\frac{1}{|T|} \sum_{t_i = t_0}^{t_n} p^j_{err} \int_{-\epsilon}^{\epsilon} P^j_\epsilon(\epsilon) d\epsilon] > \omega$

#Simple case 1: Unreliable machine, reliable machine, T tasks, no re-execution
  Assume, $M = m_0$ and $c_0 < 1, c_r = 1$. The minimization problem becomes run as many tasks on the unreliable machine.

$[\frac{1}{|T|}\sum_{t_i=t_0}^{t_n} p_{err}^j \int_{-\epsilon}^{\epsilon} P_\epsilon^j(\epsilon)d\epsilon] > \omega$

We can flip the problem to consider the probability of a bad error occuring:

$[\frac{1}{|T|}\sum_{t_i=t_0}^{t_n} p_{err}^j (1 - \int_{-\epsilon}^{\epsilon} P_\epsilon^j(\epsilon)d\epsilon)] < 1 - \omega$

lets say we execute k tasks on the unreliable processor:

$[\frac{k}{|T|} p_{err}^j (1 - \int_{-\epsilon}^{\epsilon} P_\epsilon^j(\epsilon)d\epsilon)] < 1 - \omega$

$k < \frac{|T|(1-\omega)}{p_{err}^j (1-\int_{-\epsilon}^{\epsilon} P_\epsilon^j(\epsilon)d\epsilon)}$

We can run a maximum of k tasks on the unreliable machine where k is

$k < \frac{|T|(1-\omega)}{p_{err}^j (1-\int_{-\epsilon}^{\epsilon} P_\epsilon^j(\epsilon)d\epsilon)}$

1.5 Outlier Detector

Suppose we have an outlier detector $P_{out}(\vec{r})$:

if $P_{out}(\vec{r}) \neq 0 \rightarrow ok$

else $\rightarrow outlier$

Outlier Detector

Suppose the outlier detector perfectly captures the reliable result probability distribution:

$P_{out}(\vec{r}) = P(\vec{r}) = \int_i P(\vec{r}, \vec{i}) P(\vec{i}) di$

Impact on Error

The maximum error is bounded by

$\epsilon_{max} = argmax_{\vec{r}_a, \vec{r}_b} \quad \sqrt{(\vec{r}_a - \vec{r}_b)^2} \quad | \quad P_{out}(\vec{r}_a) > 0 \;\; and \;\; P(\vec{r}_b) > 0$

Let $f_e(\epsilon) = \{1 \;\; when \;\; |\epsilon| \leq \epsilon_{max}, 0 \;\; otherwise\}$.

The new error distribution with this outlier detector is:

$P_e^{j\prime}(\epsilon) = P_e^j(\epsilon) \cdot f_e^j(\epsilon)$

Impact on Cost

The probability of re-execution is

let $g_e(\vec{r}) = \{0 \;\; if \;\; P_{out}(\vec{r}) > 0, \;\; 1 \;\; otherwise\}$

$p_{reexec}^j = \int_{-\infty}^{\infty} Pj'(\vec{r}) \cdot g_e(\vec{r})$

So the added predicted cost for machine j is

$c_r \cdot p_{reexec}^j + c_j$

1.6 Martin Notes

```
================================================
Add probability outlier detector detects outlier (increase cost probabilistically,
    improve accuracy probabilistically)
=================================================
        Martin:
        * 2 different probability distributions, each output comes from a mixture of
             the two. (statistical inference,           figure out likelihood) error
```

```
     prob distribution, and non-error distribution
optimization problem: pay cost to get truth.

* joint probability distribution of the inputs and the outputs (
    deterministic). Project the dots down
onto the outputs (integration), figure out which distribution the output
    comes from.
```

2 Model

Machine Model
Worker Machine $m \in M, |M|$
Main machine R
Each worker machine has cost:
$c = a \cdot work \cdot t + b \cdot energy \cdot t$

energy: energy per time.
work: amount of work per time $t = t_a, t_b$.
For each task T, the machine has the following properties:
For (m, T)
P_{crash}: The per-task runtime probability distribution.
P_{hang}: The per-task runtime probability distribution.
$P_{r_i}(x|r, t)$: The probability distribution of results produced by this machine, given the correct output r and the runtime.
Note: These properties are not known by topaz

Programming Model
For taskset T: $t \in T, |T|$
Task inputs and outputs: $r : R^n \to R^m \mid t(i_1, i_2 \cdots) \to (r_1, r_2, \cdots)$
The inputs are real values. A fixed array of size n is interpreted as n unit values.
each task has a runtime: t

Observed Properties

For each machine:
For $(m, T)|m \in M \to (\{OD_{r_i}\}, OD_{time})$
each task has a time that it takes to execute on some machine $\hat{P}_{r_i}(x)$: For each result r_i, the observed output probability distribution. Used to determine if a runtime is reasonable.
$\hat{P}_{time}$: The observed distribution of runtimes. Used to determine if a runtime is reasonable.

Scheduling Tasks
$t_1, t_2, ... \to m$

topaz may send any number of tasks at once to machine m.

3 Programming Model

4 Notation

Tasks: $t_i \in T$, $|T| = n$

Machine: $m_j \in M, |M| = m$, $r : reliable$

Cost: $m_j^w \in M \mid c_j = a \cdot \frac{1}{speed_j} + b \cdot energy_j$

Martin: each machine has a speed and energy. Each task has an amount of work. The amount of time it takes machine to execute task i is $work_i \cdot speed_j$*.*
Topaz executes on a distributed system with m worker machines, $m_i^w \in M$, $|M| = m$ and a reliable main machine m^r. Each worker machine has a cost associated with it c_i, the reliable machine has a cost c_r.
Optimization Problem: we want to minimize the cost of the execution of the taskset, C, while meeting the quality of result, Q.

4.1 Cost

The cost of a machine $c_i = f(perf, energy)$, where $c_i \propto \frac{1}{perf}$ and $c_i \propto energy$. We can express $c_i = a \cdot \frac{1}{perf_i} + b \cdot energy_i$, where the weights may be user specified. I separate the cost from properties of the machine to keep the cost model sufficiently abstract.

Assumption: *The reliable machine is more expensive than any of the other machines in the cluster* $\forall c_i : m_i \in M; c_r > c_i$

4.2 Correctness

We need some way of quantifying the quality of the result. Given a particular solution, each machine generates a result from a distribution centered around the solution. For example, the reliable machine generates the solution with $P(Sol) = 1$. For now on s_i is the correct solution for task i.

4.2.1 Data Parallel Problems

for problems where each task generates an independent piece of data (ex/pixels in an image), we wish to enforce the following constraint:

Strong Assertion: All tasks meet a result constraint.
$\forall t_i \in T, t_i \to m_j \mid (\int_{-\epsilon}^{\epsilon} P_j(s_i + \epsilon)) > P_{spec}$
ex: black-scholes, we want to ensure each price is likely to give a correct answer

Weak Assertion: On average, all tasks meet a result constraint.
$t_i \to m_j \mid \frac{1}{n}\sum_{i=1}^{n}(\int_{-\epsilon}^{\epsilon} P_j(s_i + \epsilon)) > P_{spec}$
ex: image generation, allow individual pixels to be arbitrarily bad as long as entire image is on average, ok.

Issues: we don't know the error distribution relative to the solution beforehand, we would need to build a per-machine error distribution during runtime. We can empirically estimate $1 - \int_{-\epsilon}^{\epsilon} P_j(s_i + \epsilon))$ at runtime by re-executing tasks at runtime and taking the number of incorrect results over the total number of results. We will call this p_j^f, the probability of task failure for machine j.

4.3 Assumptions

Martin: No batching then batching. Bad machine generates fault result U(-infty, infty), actual result is N(0,1)

- *No Batching*: one task execution on one machine, followed by one reduction.
- *No Catastrophic Failures*: Typically if you send too many tasks to a machine, it may crash. We won't model this for now.
- *Per-Item Outlier Detection*: Simple per-primitive outlier detector - no cross-item correlation.
- *User provides reliability specification*: user provides P_{spec} and ϵ for each data element in each task set.

4.4 Parameter Space

Topaz has the following parameters to play with.

Scheduling: [cost:c_j] We can choose $j : t_i \to m_j, m_r$:
Outlier Detection: We can configure different parameters of the online outlier detector. For our basic outlier detector, that is the number of standard deviation we can use.

5 Naive Model

More Assumptions:

- *We're discussing a taskset with a single numerial output.*
- *Output Distribution follows Normal Distribution*: A simplification.
- *Data Parallel, Strong Specification*: provide a hard desired probability p_{spec} falls within ϵ

5.0.1 Scheduling

We greedily choose the smallest c_j where $1 - p_j^f$ (the probability of a particular element being outside of $s_i \pm \epsilon$) meets p_{spec}

5.0.2 Outlier Detector

Online detector that models the output distribution as $N(\mu, \sigma)$. This does not work well for output distributions that do not resemble normal distributions. If an outlier detector detects an outlier, for now, always re-execute.
Re-execution: [cost:$c_j + c_r$] We can get a completely correct result and improve the outlier detector on outlier detection.
Parameters:

- K $\rightarrow$ the number of standard deviations from mean to consider ok.

Martin analysis: lets take a particular computation where we add up all the numbers first. Outlier detector algorithm (normal distribution), Prove some bound on how results are likely to be. Martin: we will eventually need to include task work - lets leave task work out. Issues we will eventually have to deal with.

- Amount of work in each task (ie tasks have variable work quantities)
- Batching
- Crashes - outlier detection on task times
- 2 kinds of re-executions: pitch result and re-execute. execute someplace else.

Outlier detection strategies: 1. main processor can send outlier data to child machine to remote machine.

6 Example: Array Summation

Assume

- we $\sum$ the results of T tasks, where $t : i \rightarrow r$ and $r \in N(\mu, \sigma)$.
- we have an outlier detector $N(\mu, k\sigma)$ that predicts the output distribution and considers results within k standard deviations ok, where k is unknown.
- on outlier detect, we re-execute
- we have a desired percent error of p
- assume w percent of tasks fail, Assume the output distribution is uniformly distributed $U(F_{min}, F_{max})$.

$p = \frac{S_{urel} - S_{rel}}{S_{rel}} = \frac{\sum r_{urel} - r_{rel}}{\sum r_{rel}}$
All undetected outliers fall within $(\mu - k\sigma, \mu + k\sigma)$. In the worst case: $N(\mu, k\sigma) \rightarrow |r_{urel} - r_{rel}| \sim 2k\sigma$
With increasing k, the percent of undetected outliers increases:
$\int_{\mu-k\sigma}^{\mu+k\sigma} \frac{1}{F_{max} - F_{min}}$
$[(\mu + k\sigma - \mu + k\sigma)\frac{1}{F_{max} - F_{min}}]$
$[\frac{2k\sigma}{F_{max} - F_{min}}]$

We can approximate the absolute error as follows:
$\sum r_{urel} - r_{rel} \approx w \cdot |T| \cdot \frac{2k\sigma}{F_{max} - F_{min}} \cdot k\sigma = w \cdot |T| \cdot \frac{2k^2\sigma^2}{F_{max} - F_{min}}$
We can approximate the average sum as follows:
$\sum r_{rel} = |T|\mu$

We rewrite p as:
$p \approx \frac{w \cdot |T| \cdot 4k^2\sigma^2}{(F_{max} - F_{min}) \cdot |T|\mu}$
$p \approx \frac{w \cdot 4k^2\sigma^2}{(F_{max} - F_{min})\mu}$
$\frac{(F_{max} - F_{min})\mu \cdot p}{w \cdot 2\sigma^2} = k^2$

Therefore, the ideal number of standard deviations to consider, given a desired output error rate, is:
$k = \sqrt{\frac{(F_{max} - F_{min})\mu \cdot p}{w \cdot 4\sigma^2}}$

Obviously (1) we don't know the error distribution, nor does (2) our outlier detector perfectly fit the output distribution. The the rate of task errors for a particular machine is not known.