TEXAS ADVANCED COMPUTING CENTER
WWW.TACC.UTEXAS.EDU

TEXAS
The University of Texas at Austin

A Simple Cost-model for Comparing Diverse Architectures

R. Todd Evans

Texas Advanced Computing Center
The University of Texas at Austin

SC' 21, IXPUG BoF
Nov 17, 2021

How to Compare Diverse Hardware?

Diverse hardware has wide-ranging costs and benefits.

- NVIDIA A100 costs more than RTX but is "faster"
- A100 costs more than Intel Platinum but is "faster"
- Is the extra cost worth it? Depends on workload and how fast you need the results (*result depreciation*).

How to compare Costs vs Benefits of hardware choices?

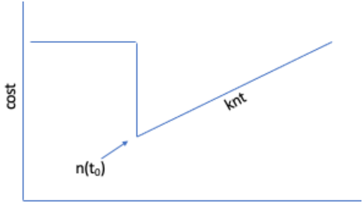
- Capture costs in Total Cost to Own (TCO) architecture i per unit time as κ_i
  - Depreciation of hardware or Cloud Provider charges per hardware device
  - Power and cooling cost per device
  - Other costs that increase with device count
- Benefits
  - Reduced result depreciation (more productivity/throughput, e.g. less time to science)

Cost Model: Cost of a Run

Find minimum cost hardware for a **run**

- run = figure of merit for an application (one iteration, ns, time-to-solution)
  - *cost-per-run* $= c$ [\$]
- $\kappa_i =$ holistic TCO of 1 device of architecture i per unit time
- assume portion of workload benefits from parallelization
  - $t_i = t_{i,s} + t_{i,p}/n_i$ (measure $t_{i,s}$, $t_{i,p}$ in a scaling study)
  - $c_i = \kappa_i n_i t_i = \kappa_i (n_i t_{i,s} + t_{i,p})$
  - $t_{i,s}$, $t_{i,p}$ are fixed for a given architecture and workload $\rightarrow$ c_i strictly increasing with n_i
- $n_i = 1$ is always c_{min} ("always" a serial component)
- Minimizing cost-per-run maximizes runs/$ (throughput) !!!

Cost Model: Strict Performance Requirements



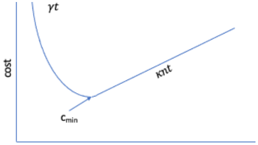
Most cost effective architecture *i* is straightforward with rigid requirements

- Big cost penalty for not completing run in time t_0
- $c_i = \kappa_i n_i t_i = \kappa_i(n_i t_{i,s} + t_{i,p})$ were $t_i \leq t_0$ is required
- If serial workload then $n_i = 1$ and $c_{min} = min(\{\kappa_i t_{i,s}\})$
- if parallel workload then $n_i = \frac{t_{i,p}}{t_0 - t_{i,s}}$ and $c_{min} = min(\{\kappa_i \frac{t_{i,p} t_0}{t_0 - t_{i,s}}\})$
  - Memory footprint requirements work similarly

Cost Model: Relaxed Performance Requirements

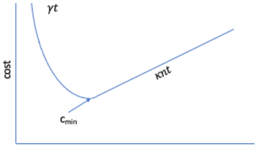
Relax performance requirements but account for depreciation of the run result

- If no penalty for longer runtime than always run serially ($n_i = 1$)
- Computational result is worth less in the future than it is *now*
  - value of computational results depreciate over time
- Assume linear depreciation of result with time: $V(t) = V_0 - \frac{V_0}{T} t = V_0 - \gamma t$



  - $c_i(n_i) = \underbrace{\kappa_i n_i t_i(n_i)}_{compute\ cost} + \underbrace{\gamma t_i(n_i)}_{result\ depreciation}$
- *result depreciation* difficult to quantify but lower runtime *is* valued
- *compute cost* is off-set by *result depreciation*

Cost Model: Minimize $c_i = \kappa_i n_i t_i + \gamma t_i$



Minimize cost-per-run of architecture i: c_i

- $dc_i/dn_i = 0$
  - $n_i = \sqrt{\frac{\gamma t_{p,i}}{\kappa_i t_{s,i}}}$ (cost-optimal number of devices to use)
- c_{min} for architecture i
  - $c_{min,i} = \left(\sqrt{\gamma t_{s,i}} + \sqrt{\kappa_i t_{p,i}}\right)^2$
- $c_{min} = min(\{c_{min,i}\})$

Cost Model: Estimating γ?

What is the depreciation rate (γ) of a run's result?

- If γ and fraction of work parallelizable are equal for each architecture then *relative* costs are independent of γ
  - Choose the cheapest
- Run results correlate to R&D output or revenue
  - images recognized, search results returned, bitcoin mined, ns simulated etc.
- If related to revenue γ may be straightforward to estimate
  - lost asset appreciation (e.g. invest V_0 now and make γt interest on a loan)
  - drug revenue per unit time
- If related to pure γ R&D (e.g. TACC) probably more abstract
  - Assume R&D value is as valuable as the HPC infrastructure that supports it.
  - Include any cost that doesn't increase with device count
    - total system budget
    - cost to port to new architecture (γ_i), licenses

Summary

Demonstration of Cost-model to Optimize a System Procurement

*Evans R.T. et al. (2021) Optimizing GPU-Enhanced HPC System and Cloud Procurements for Scientific Workloads. In: Chamberlain B.L., Varbanescu AL., Ltaief H., Luszczek P. (eds) High Performance Computing. ISC High Performance 2021. Lecture Notes in Computer Science, vol 12728. Springer, Cham.

Many Uncertainties

- Amdahl's law approximations introduce uncertainties. We do know c_{min}
  - not in super-linear scaling regime
  - not in regime where scaling fails
  - can add terms to t_i (e.g. $log(n_i)$) and solve transcendental for n_i
- Defining proxy workload
- Depreciation not necessarily linear
- Estimating TCO - κ_i - precisely is exhausting
  - Difficult to measure: Can be quite difficult if taking code porting into account
- γ can be tricky - consider lost revenue or cost of system

Thank you

R. Todd Evans
HPC Manager, P&A Group
Texas Advanced Computing Center
University of Texas at Austin
rtevans@tacc.utexas.edu