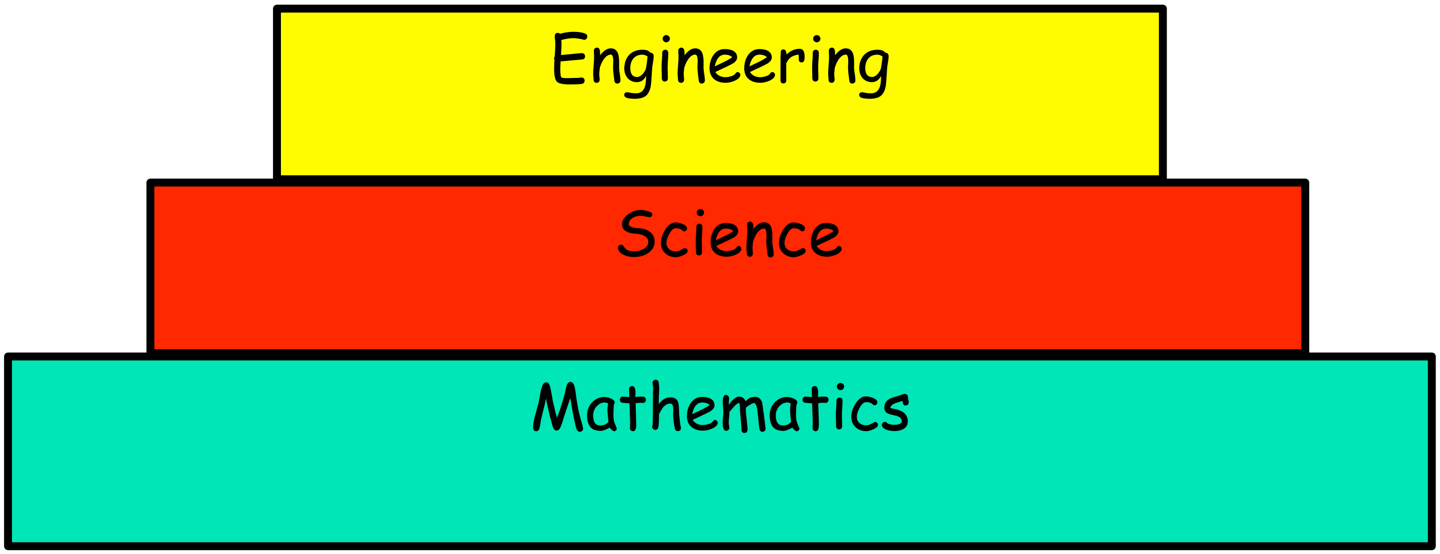
On the Science of Speed Scaling From an Algorithmist's Viewpoint

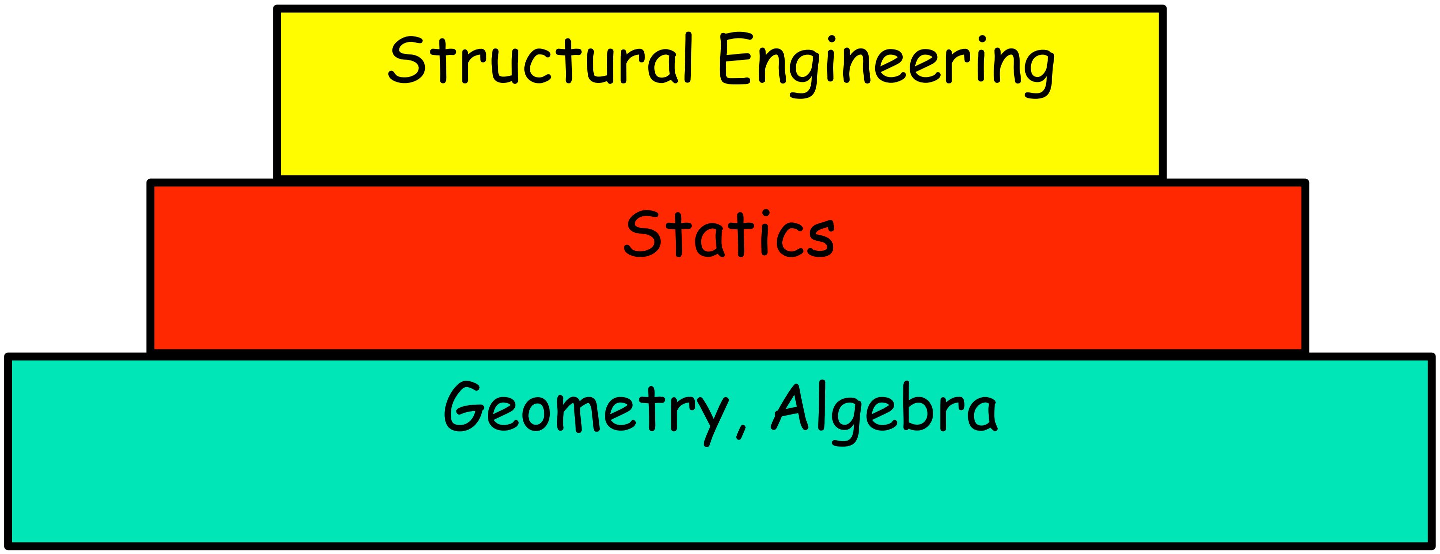
Kirk Pruhs

University of Pittsburgh

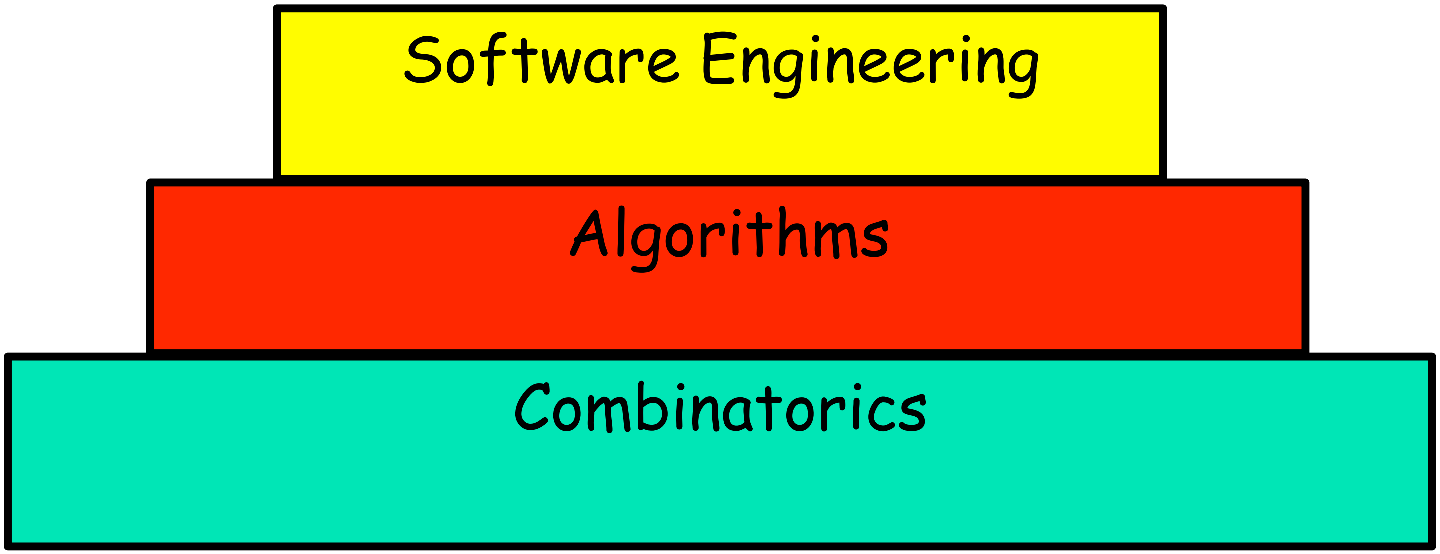
Science's Place in the World



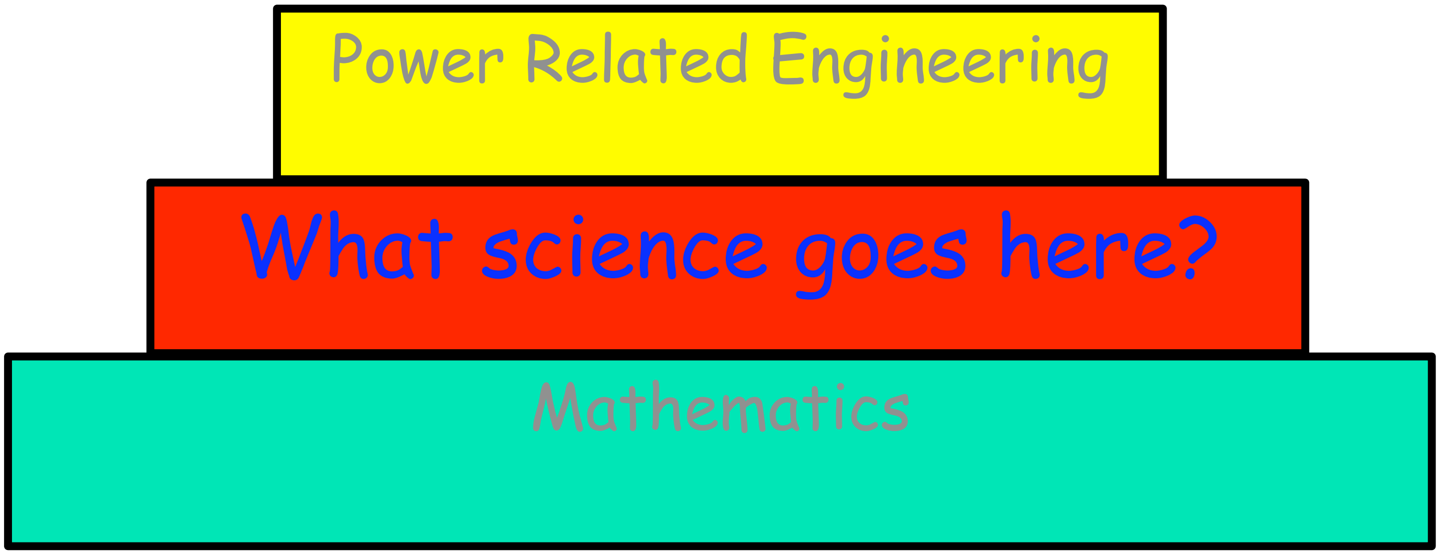
Science's Place in the World



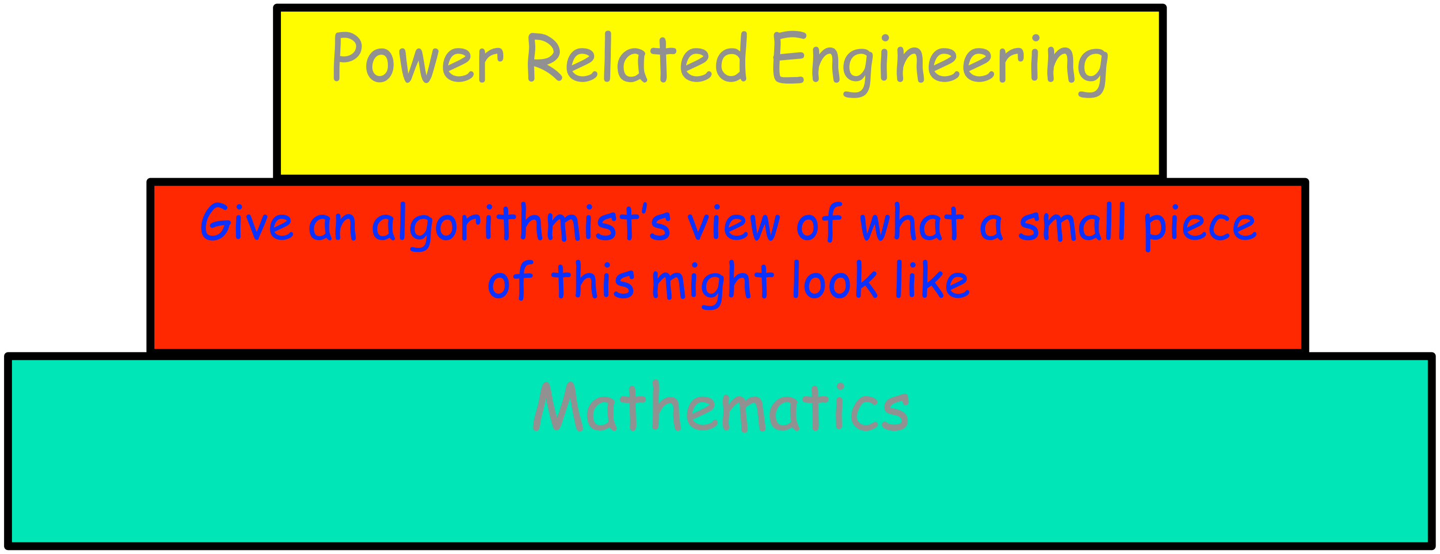
Science's Place in the World



Workshop's Goal: Give Some Initial Answer

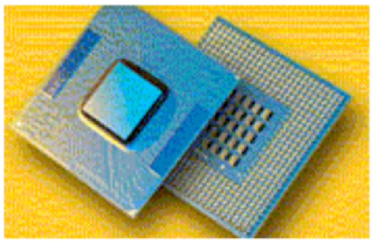


This Talk's Goal



Motivating Technology: Speed Scaling

Mobile Intel® Pentium® 4 Processor - M



Built on 0.13-micron process technology and Intel® NetBurst™ microarchitecture, the Mobile Intel® Pentium® 4 Processor - M provides innovative capabilities for graphics-intensive multimedia applications. It's also excellent for processor-intensive background computing tasks, such as compression, encryption, and virus scanning.

Enhanced Intel SpeedStep® technology helps to optimize application performance and power consumption, and Deeper Sleep Alert State, a dynamic power management mode, adjusts voltage during brief periods of inactivity—even between keystrokes—for longer battery life.

Mobile Intel® Pentium® 4 Processor – M Features

Available Speeds	**2.60 GHz**, 2.50 GHz, 2.40 GHz, 2.20 GHz, 2.0 GHz, 1.80 GHz, 1.70 GHz, 1.60 GHz, 1.50 GHz, 1.40 GHz
Chipset	Mobile Intel® 845 Chipset Family
Cache	512 KB On-Die Level 2 (L2) Cache
RAM	up to 1GB DDR SDRAM
System Frequency Bus	400 MHz

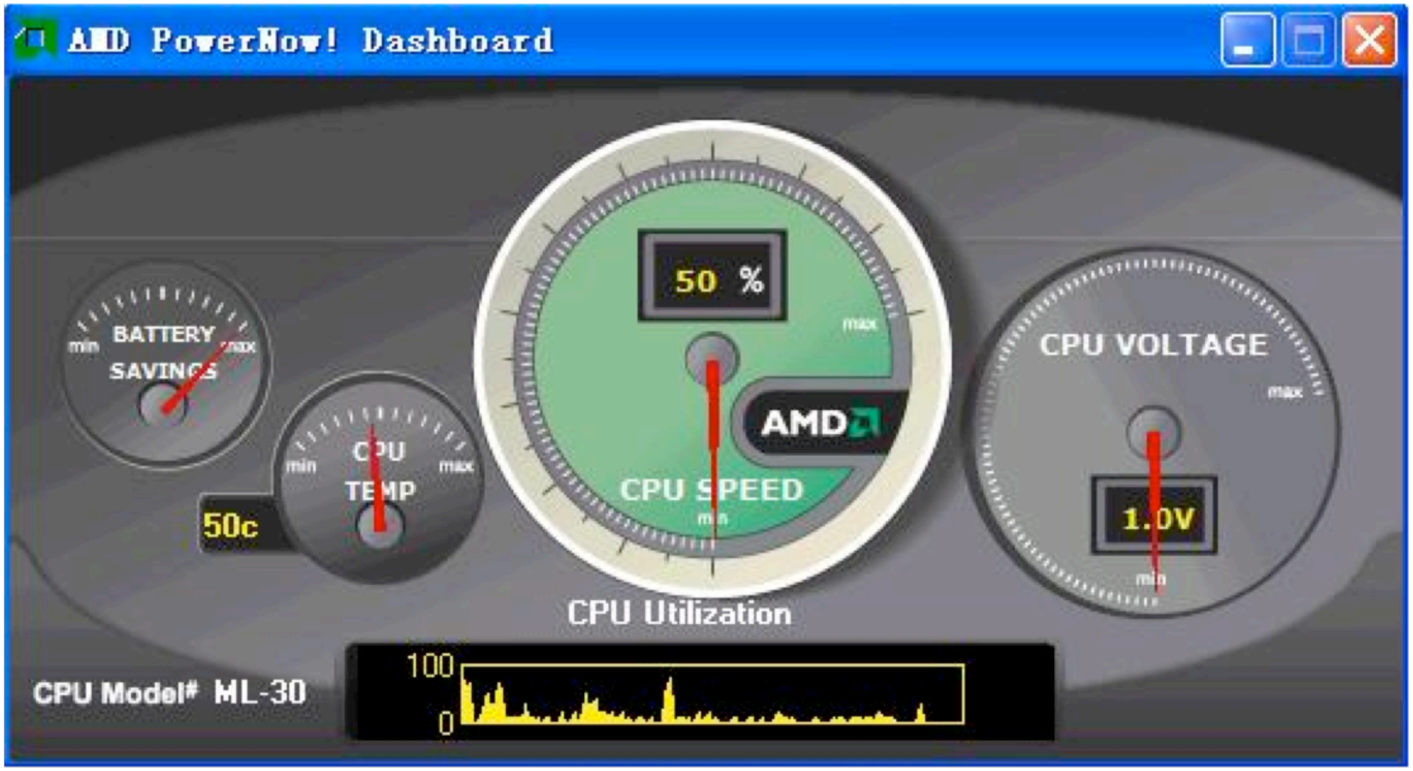
Product
- Processor
- Specs Up
- Datasheet
- Performan
- Applicatio
- Design Gu
- Frequently
- Processor
- Technical
- Boxed Mo Processor

Whe

Tools
- Find the R
- Intel® Pro Benchmar
- Compare

Technic

Motivating Technology: Speed Scaling



One Question Raised By This Technology

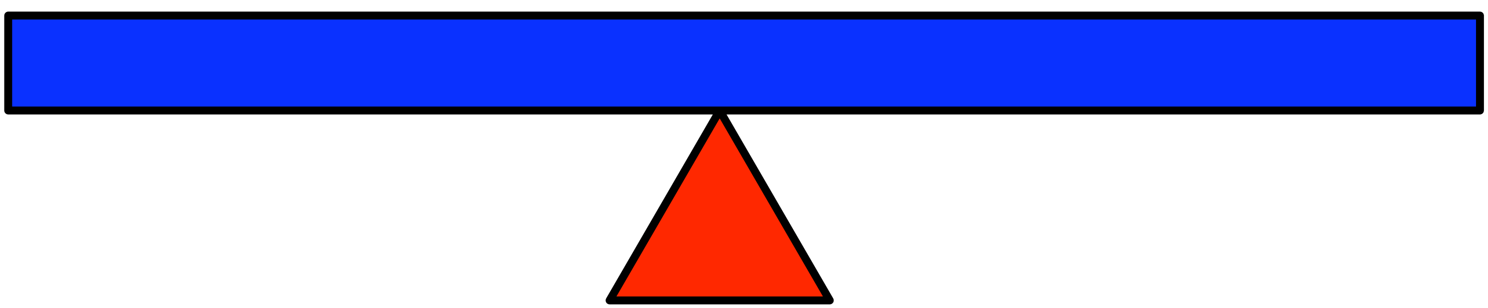
- Engineering Question: How do you manage/ scale the speed/power, and how does this interact with scheduling?
  - Overarching engineering question for this talk

Science from the Algorithmist's View

- Science research tries to model a complex system by something simple, accurate, amenable to math and predictive.
  - Google researcher Muthu Muthukrishnan's "My Slice of Pizza" blog

- Accurate
- Realistic
- Predictive

- Simple
- Amenable to math



Another Muthu Quote:

- I realized this week in a meeting (at Google) that we have people who see the tree and those that see the forest, neither is useful. We need people who see a tree and know it is in a forest, and who see a forest and know that it is made up of trees.

Agenda

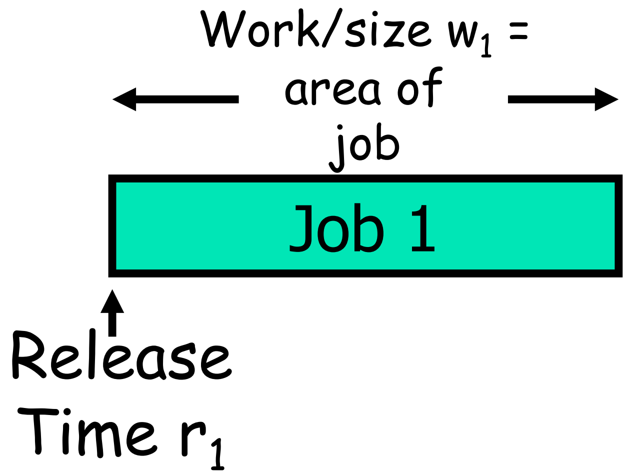
- Forest: How do you manage/scale the speed/power, and how does this interact with scheduling?
- Enumerate some of the interesting trees/models in the forest:
  - Physics
  - Objective
  - Analysis method
  - etc.
- Explain what is known about a couple interesting trees, and relate this back to the forest

Modeling Policies/Algorithms and Schedules

- The system needs
  - Job selection algorithm: determines which job is run
    - e.g. Shortest Job First,
    - Shortest Remaining Processing Time,
    - Shortest Elapsed Time, etc.
  - Speed scaling algorithm: determines the speed that the processor is run

Modeling Policies/Algorithms and Schedules

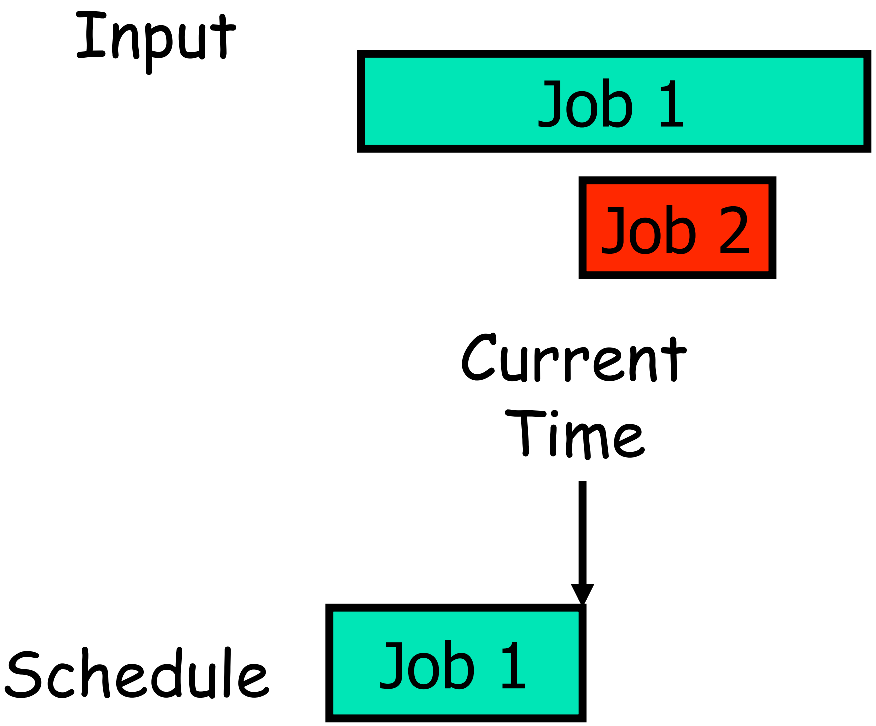
Input



Schedule

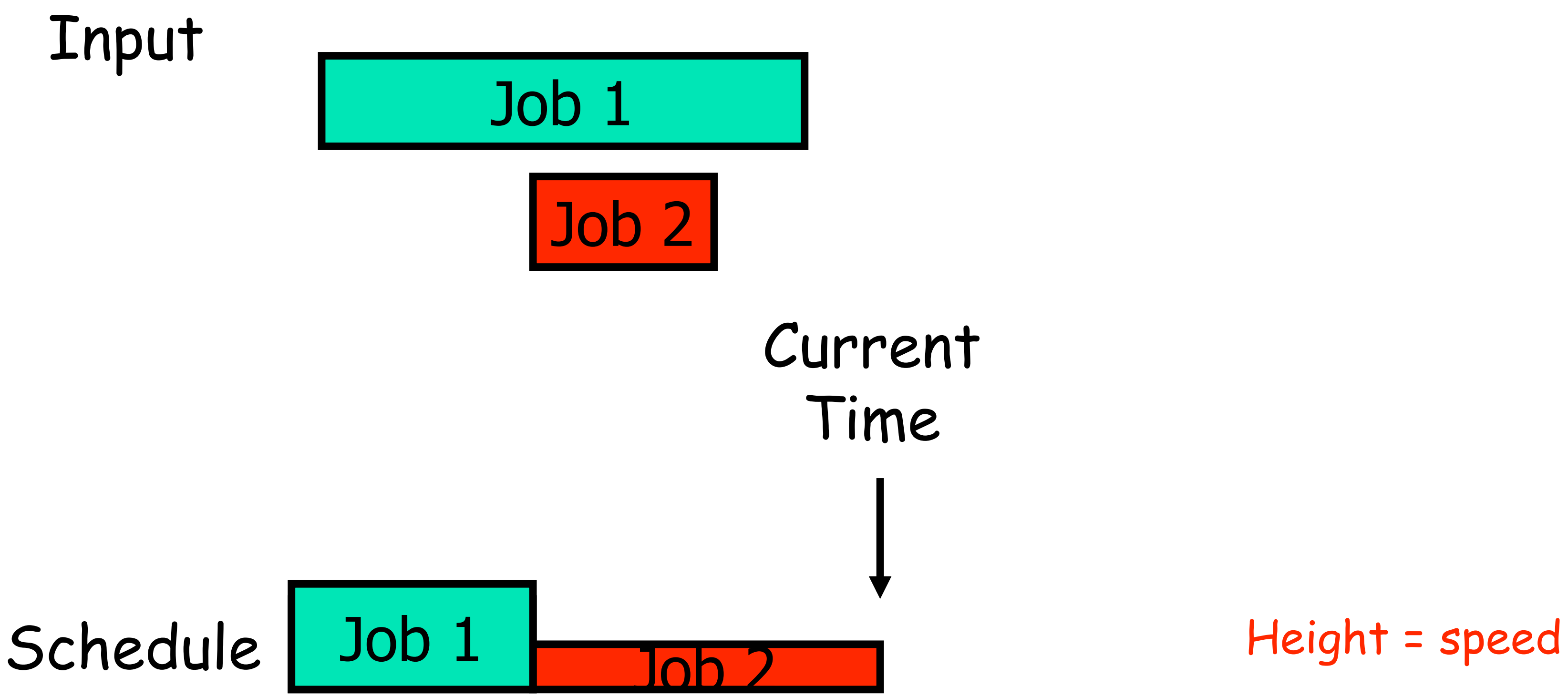
Height = speed

Modeling Policies/Algorithms and Schedules

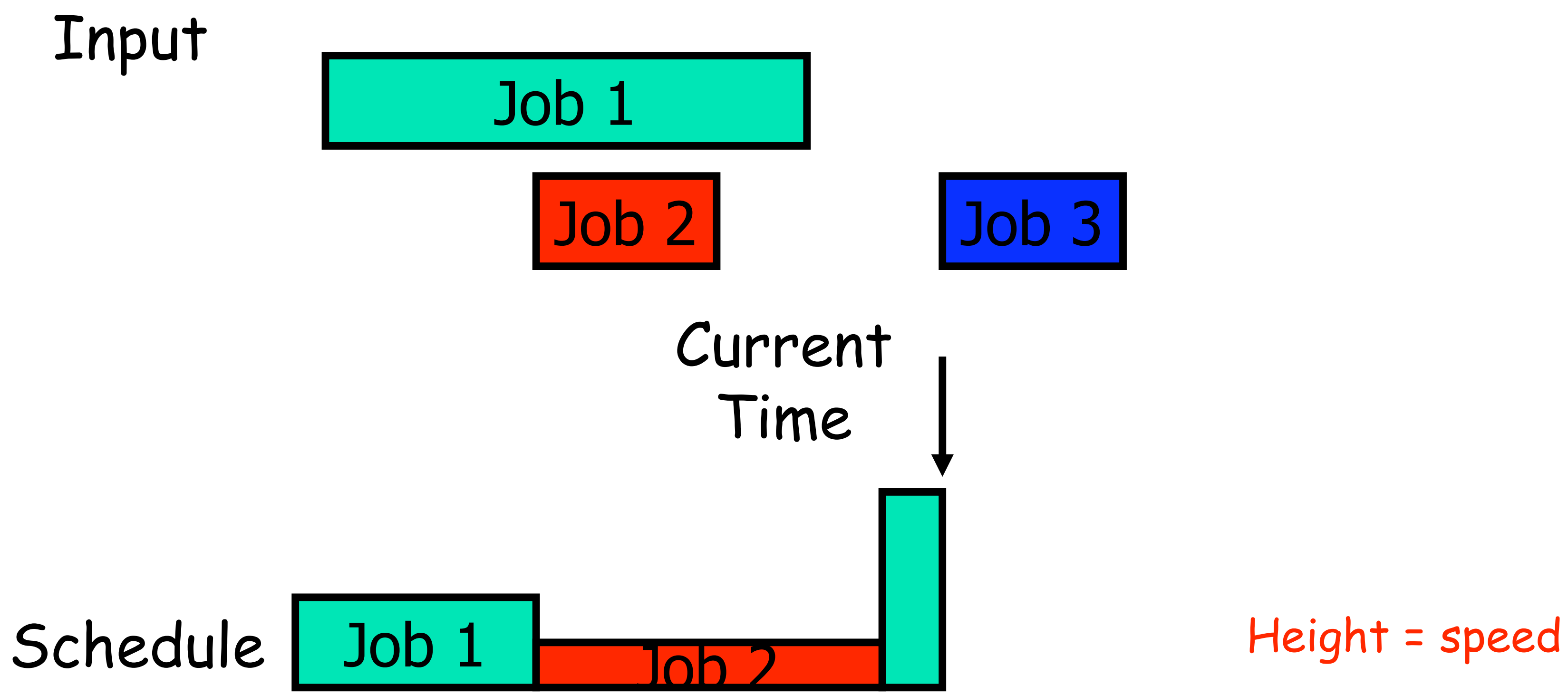


Height = speed

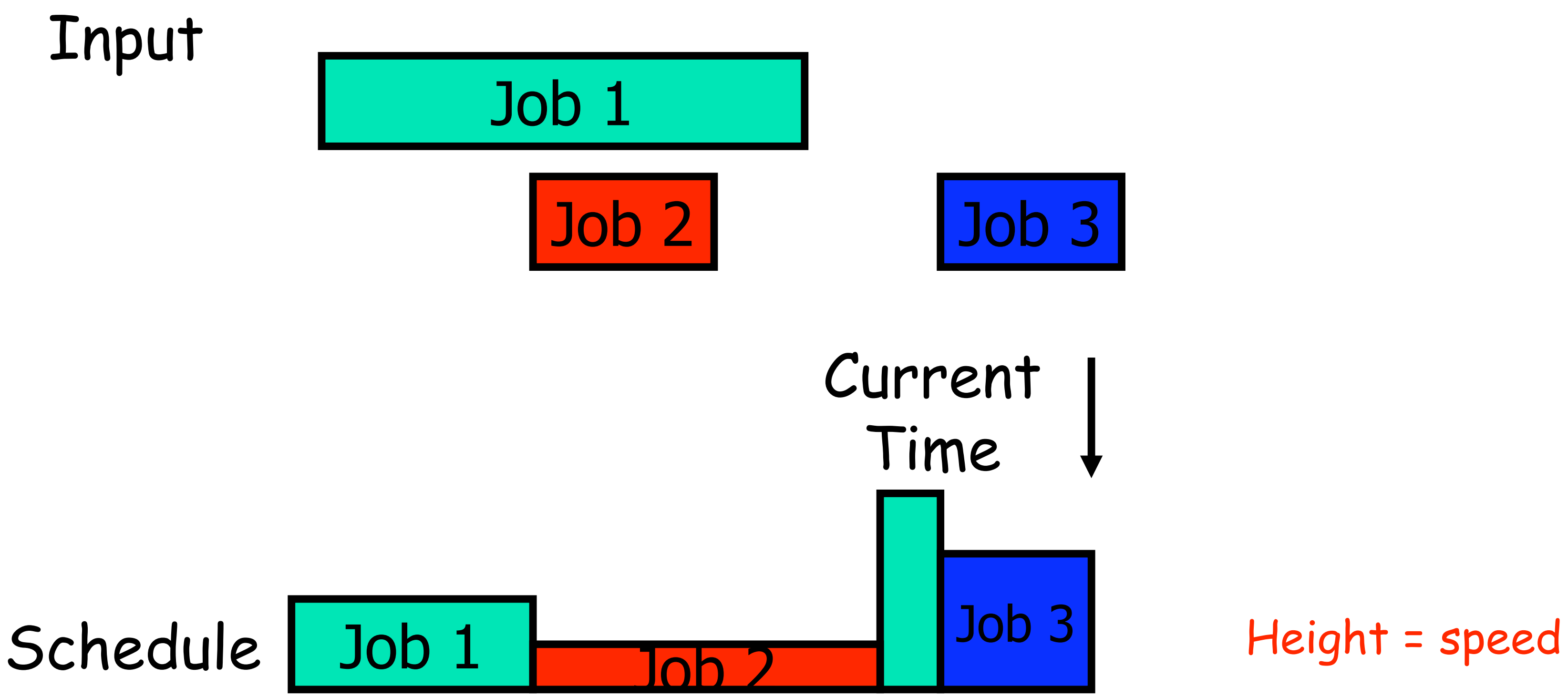
Modeling Policies/Algorithms and Schedules



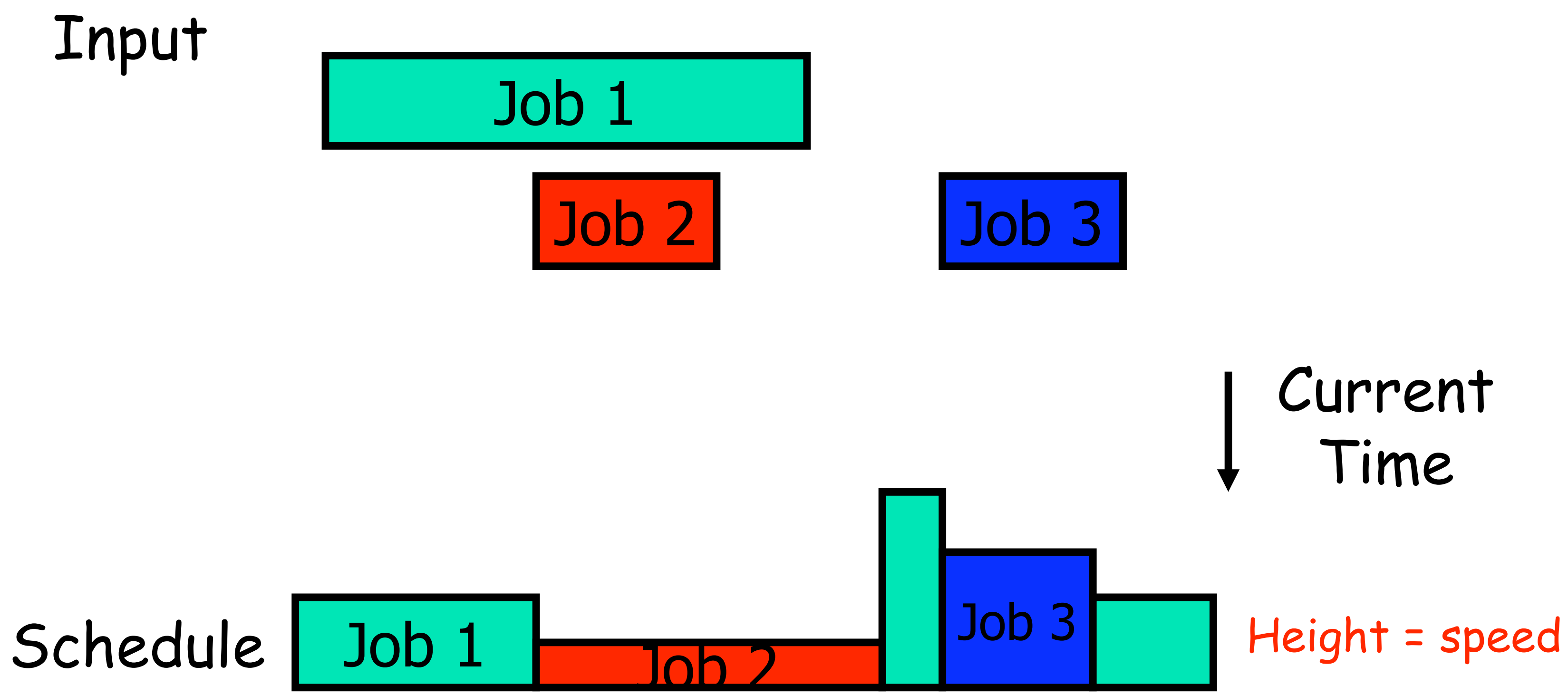
Modeling Policies/Algorithms and Schedules



Modeling Policies/Algorithms and Schedules

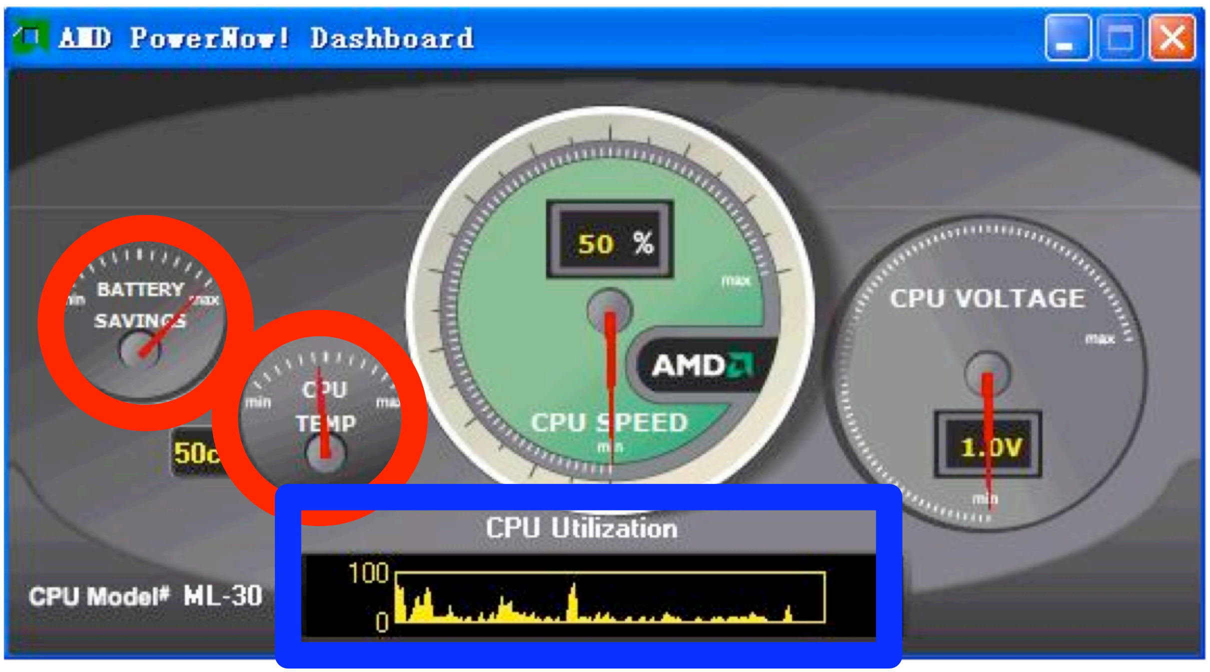


Modeling Policies/Algorithms and Schedules



Scheduling Objectives

- Dual Objectives:
  - Some Quality of Service (QoS) measure of the schedule
    - e.g. deadline feasibility, average response time, worst-case response time, average slow down, etc.
  - Some power related objective
    - e.g. temperature, energy



Physical Models

- Allowable Speeds
  - Continuous and unbounded
  - Continuous and bounded
  - Discrete
- Power P as a function of speed s
  - $P = s^{\alpha}$ where α is some constant
    - Motivated by cube-root rule for dynamic power in CMOS based processors, i.e. $\alpha \approx 3$
  - $P = s^{\alpha}$ + constant static power
  - P = f(s) for some arbitrary function f

Physical Models

- Energy = $\int_{time}$ Power
- Temperature T
  - Newton's Law of Cooling: rate of heat loss of a body is proportional to the difference in temperatures between the body and its surroundings
    - $dT/dt = P - b\,T$
    - b is device dependent constant

How to Compare Algorithms?

- Average Case (Queuing Theory)
  - Assume a mathematically tractable input distribution, e.g. Poisson arrivals and exponential job sizes, and compute expected performance of the algorithms
- Worst-case Relative Error (Competitive Analysis)
  - An algorithm *A* is competitive if it has bounded relative error
    - $\max_I |A(I) - Optimal(I)|/Optimal(I) < \infty$
  - An algorithm is optimally competitive if it has minimal worst-case relative error among all possible algorithms

Agenda

- Forest: How do you manage/scale the speed/power, and how does this interact with scheduling?
- Enumerate some of the interesting trees/models in the forest:
  - Physics
  - Objective
  - Analysis method
  - etc.
- Explain what is known about a couple interesting trees, and relate this back to the forest

Cool Tree 1: Deadline Feasibility and Temperature

- Assume that somehow a system knows a deadline for each job
  - This makes the scheduling QoS objective a constraint
  - WLOG one can use Earliest Deadline First for the job selection algorithm
  - Allows one to focus on the speed scaling algorithm
- Power objective is to minimize the maximum temperature ever reached by the device.

Cool Tree 1: Deadline Feasibility and Temperature

- Understanding Newton's Law:
  $dT/dt = P - bT$
  - If b=0, max temperature = energy
  - If $b=\infty$, max temperature = max power
  - Key theorem for analysis: Maximum temperature ≈ maximum energy over any time interval of length 1/b

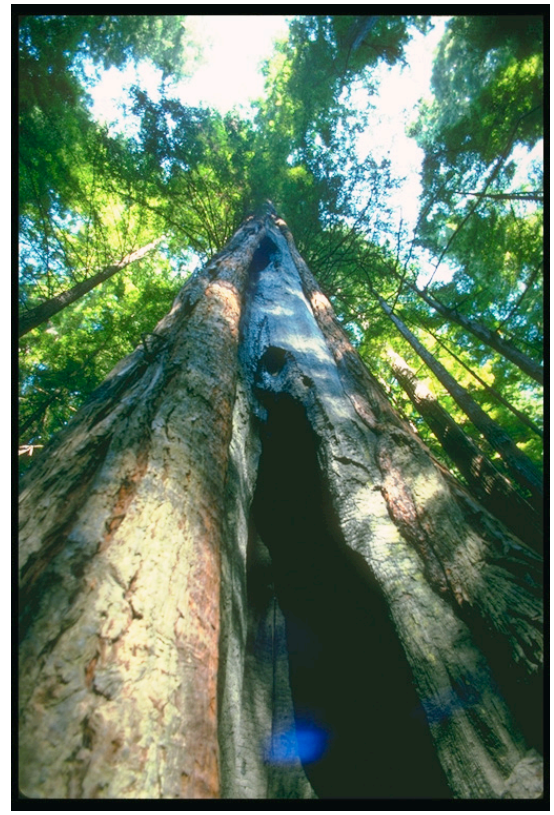
- Natural Question: Does the speed scaling algorithm require knowledge of the device specific cooling parameter b to be competitive?
  - Assume power = polynomial in speed

Cool Tree 1: Deadline Feasibility and Temperature

- Theorem: There is an optimally competitive algorithm when $b=\infty$
  - Algorithm Description: At time t, run at speed equal to the maximum over all $t_1 < t < t_2$ of $e^*w(t_1, t, t_2)$
    - Where $w(t_1, t, t_2)$ is the aggregate size of the jobs that arrive after t_1, and before t, with deadlines before t_2
  - Note that this algorithm is reasonably simple, but sufficiently non-intuitive that it is hard to imagine it being discovered by experimentation and local search
- Theorem: This algorithm is simultaneously competitive for all cooling parameters b
  - Some natural algorithms that are competitive for energy (b=0), are not competitive for larger b

Cool Tree 2: Energy/Response Tradeoff

- Setup: A user specifies an energy amount ρ that he/she is willing to spend to get a unit improvement in response time
  - e.g. I am willing to spend 3 ergs of energy for a 1 microsecond improvement in response time for a particular job
  - Response time of a job = completion time minus release time
- Resulting Objective:
  ρ *total response time + energy used



Cool Tree 2: Energy/Response Tradeoff

- Natural Question: As ρ decreases, and the total energy used decreases, can the energy of particular jobs increase in the optimal schedule ?
  - Recall ρ = amount of energy that you are willing to spend to get a unit decrease in response time

Cool Tree 2: Energy/Response Tradeoff

- Natural Question: As ρ decreases, and energy used decreases, can the energy of particular jobs increase in the optimal schedule ?
- Theorem: In the optimal schedule the power of a job J is proportional to the number of jobs that would be delayed if J is delayed (Modulo special cases)
- Answer: So as energy is lost, and jobs interfere more, this theorem forces the energy for some jobs up at a faster rate than energy is lost

- The optimal schedule doesn't change smoothly as a function of energy.
- Open algorithmic problem: find an efficient algorithm to find the optimal schedule, or proof that no such algorithm exists

Cool Tree 2: Energy/Response Tradeoff

- Natural Question: Do properties of the power function affect whether one can have a competitive algorithm?
  - Equivalently, can one reason about arbitrary power functions?

Cool Tree 2: Energy/Response Tradeoff

- Natural Question: Do properties of the power function affect whether one can have a competitive algorithm?
- One answer: If the algorithm doesn't know the size of a job when it arrives, then the algorithm can not be competitive if the power function is very steep

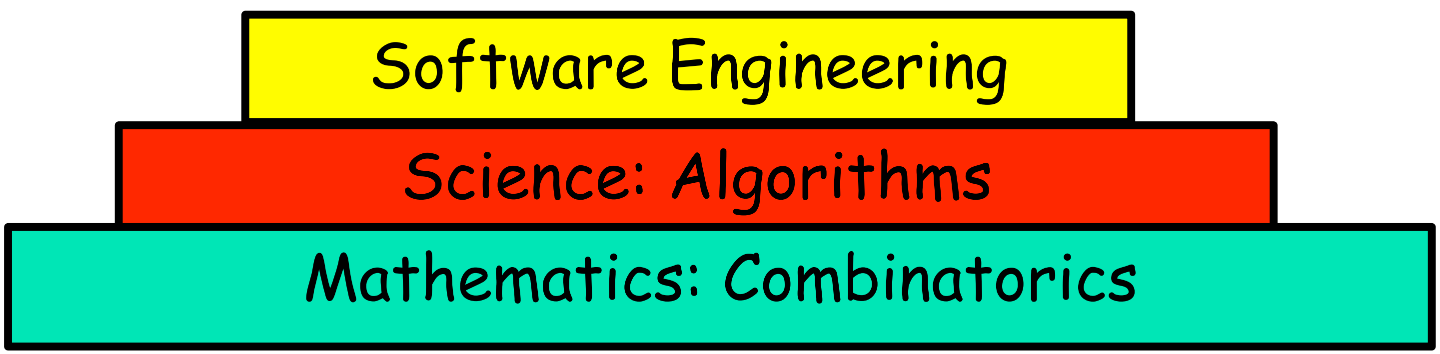
Cool Tree 2: Energy/Response Tradeoff

- Natural Question: Do properties of the power function affect whether one can have a competitive algorithm?
- Another answer: The following algorithm is competitive for all power functions:
  - Job selection: Shortest Remaining Processing Time (SRPT)
  - Speed Scaling: Power = number of unfinished jobs + 1
- The analysis requires completely different techniques than the analysis of SRPT with a fixed speed processor because
  - With a fixed speed processor the resource available per unit time is fixed
  - With a variable speed processor, the resource (energy) is global, and you get a concave nonlinear return for investing resources at a particular time

Some Concluding Remarks About the Forest:

Predictability of the Model

- Muthu's Definition: Science research tries to "model" a complex system by something simple, accurate, amenable to math and predictive.
  - In practice, predictive usually means that thinking abstractly/ mathematically within the context of the model is useful for finding a good engineering solution, not that the model perfectly predicts what will happen in practice
- Google wants software engineers that have algorithms training, not because the RAM model perfectly predicts the performance of an algorithm on a real computer, but because being able to reason about computation in an abstract model is useful when searching for a solution on a real computer

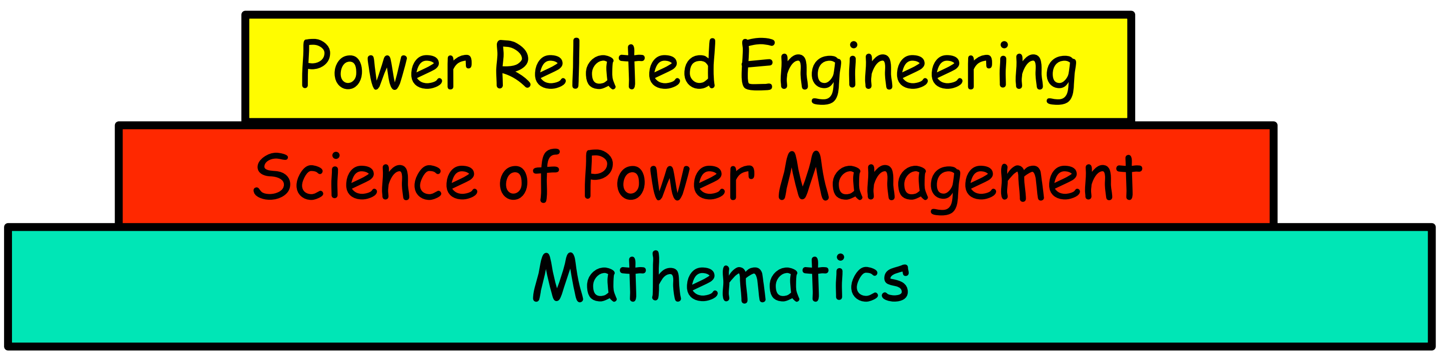


What Was Learned From This Research

- We got way better at reasoning about energy and temperature as resources, at least within the types of models considered
  - Allowable Speeds
    - Best Model: Continuous and unbounded. Other models are more complicated without providing more insight
  - Power P as a function of speed s
    - For some problems, one can reason about an arbitrary power function
  - Newton's Law of Cooling $dT/dt = P - bT$
    - The easiest way to think about approximate temperature is energy over time intervals of length $1/b$
- I can plausibly imagine teaching future engineers to think abstractly about power/energy/temperature as we currently teaching future software engineers to think about time/space.

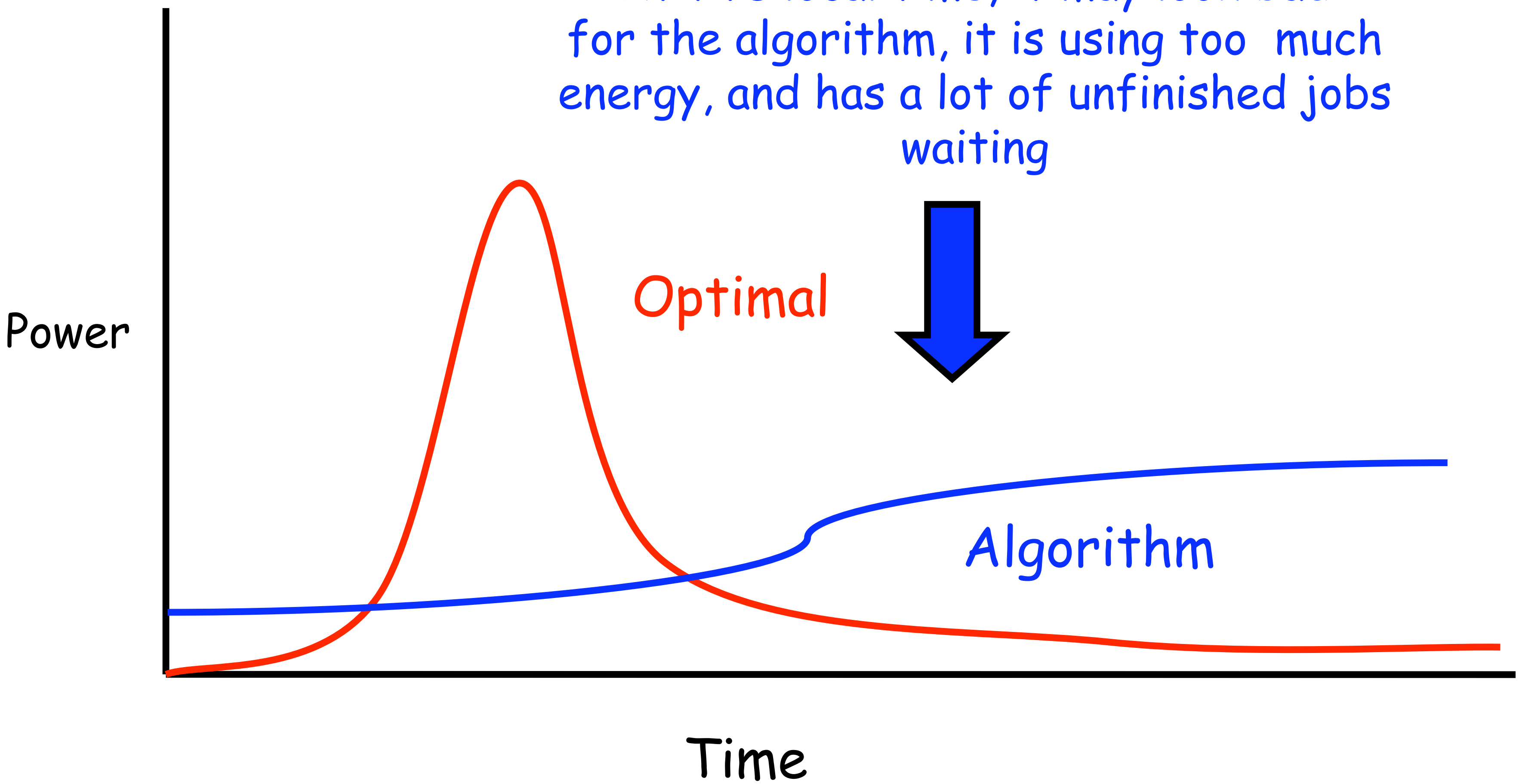
Science of Power Management

- Questions for this workshop:
  - What might a science of power management look like?
  - What areas of power management are most amenable to scientific investigation?
  - What areas of power managements would most benefit from a better scientific foundation?
  - What benefits are there for a better scientific foundation for these areas of power management?



Speed scaling analysis is hard because energy is a global resource that gives nonlinear returns from investment:

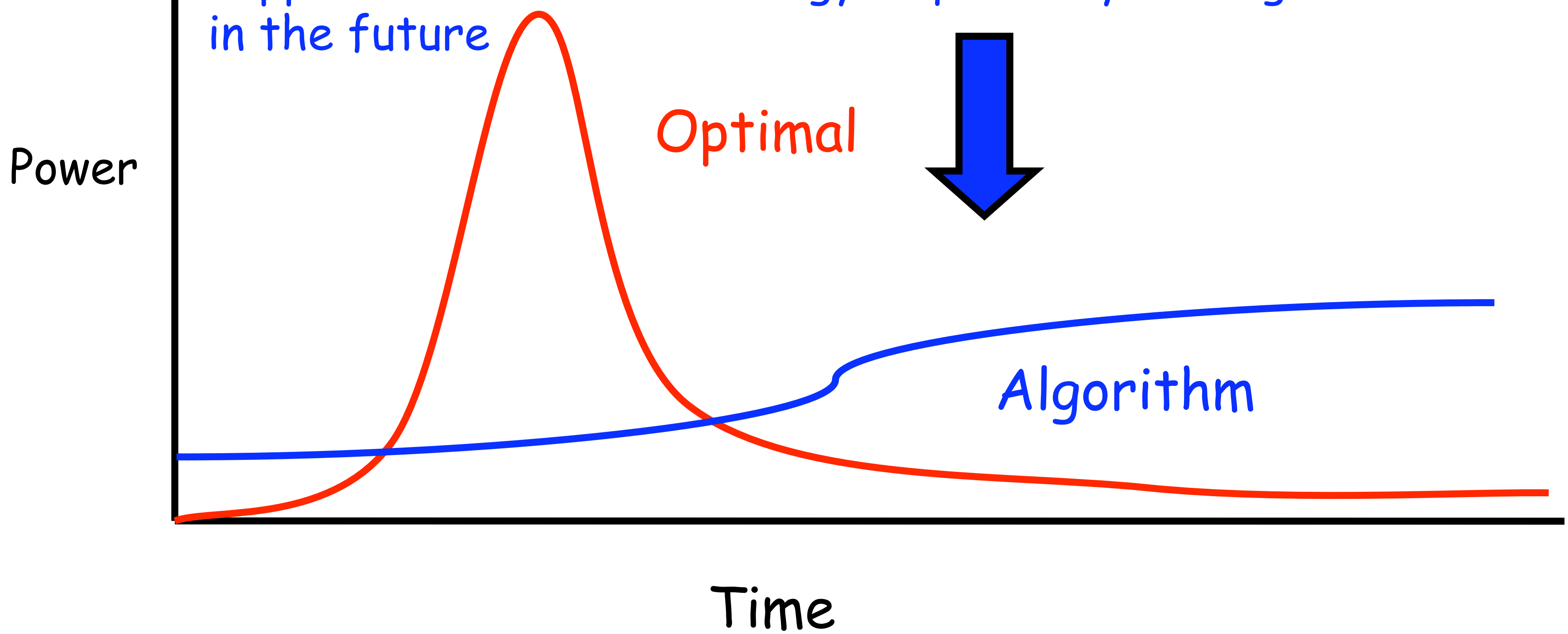
At this local time, it may look bad for the algorithm, it is using too much energy, and has a lot of unfinished jobs waiting



Speed scaling analysis is hard because energy is a global resource:

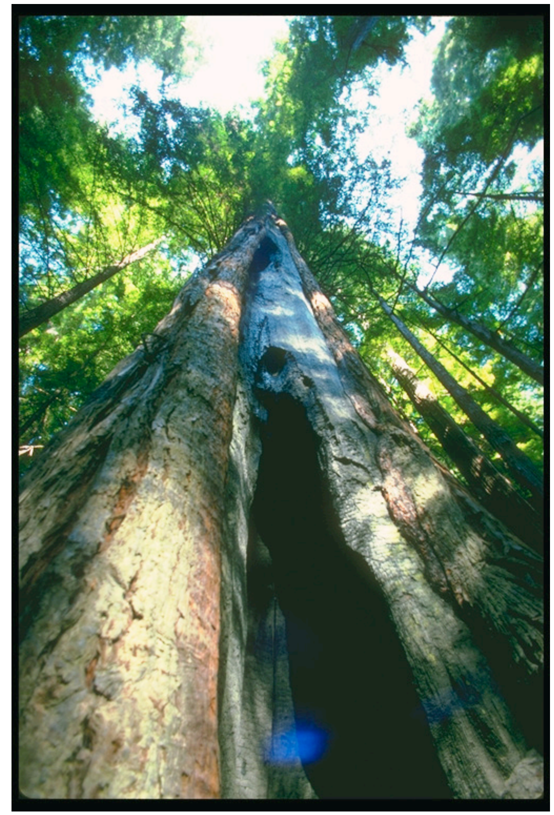
Main Analysis technique: Define a potential function Φ that:

- Lower bound on energy savings of the algorithm in the past
- Upper bound on extra energy required by the algorithm in the future



Cool Tree 2: Energy/Response Tradeoff

- If all jobs are the same size and Power=speed$^{\alpha}$:
  - the potential function Φ = (number of excess jobs)$^{2-1/\alpha}$ works
  - that is, Φ is an lower bound on energy savings in the past, and an upper bound on the extra energy required in the future



Cool Tree 2: Energy/Response Tradeoff

- For arbitrary jobs sizes and an arbitrary power function P, Φ is more complicated:
  - $\Phi = \int_{\text{job sizes } q} f(\text{number of excess jobs of size at least } q)$
  - Where $f(x) = f(x-1) + P'(P-1(x))$