Exascale Computing for Everyone: Cloud-based, Distributed and Heterogeneous

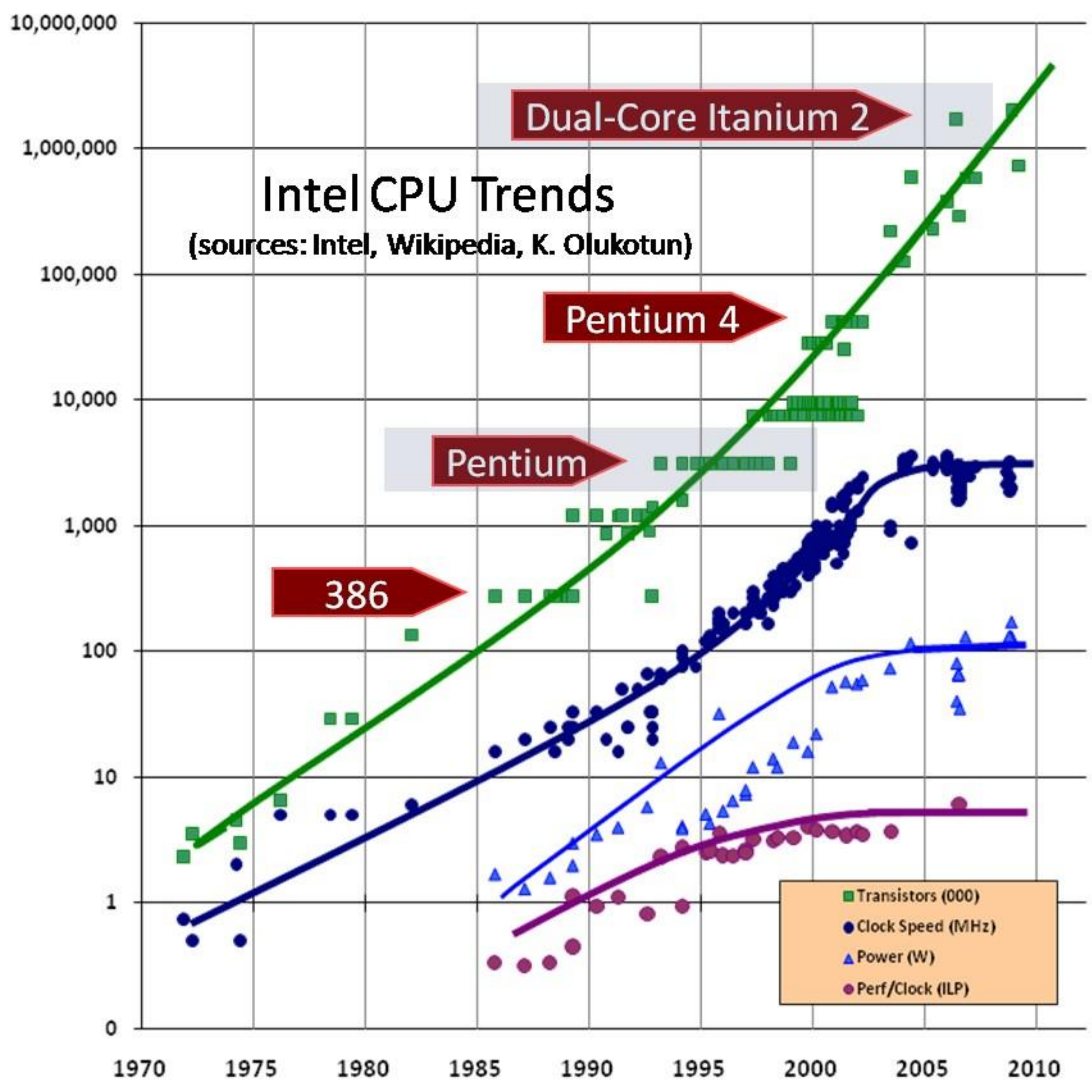
Gordon Inggs, David B. Thomas, Wayne Luk and Eddie Hung

Imperial College London

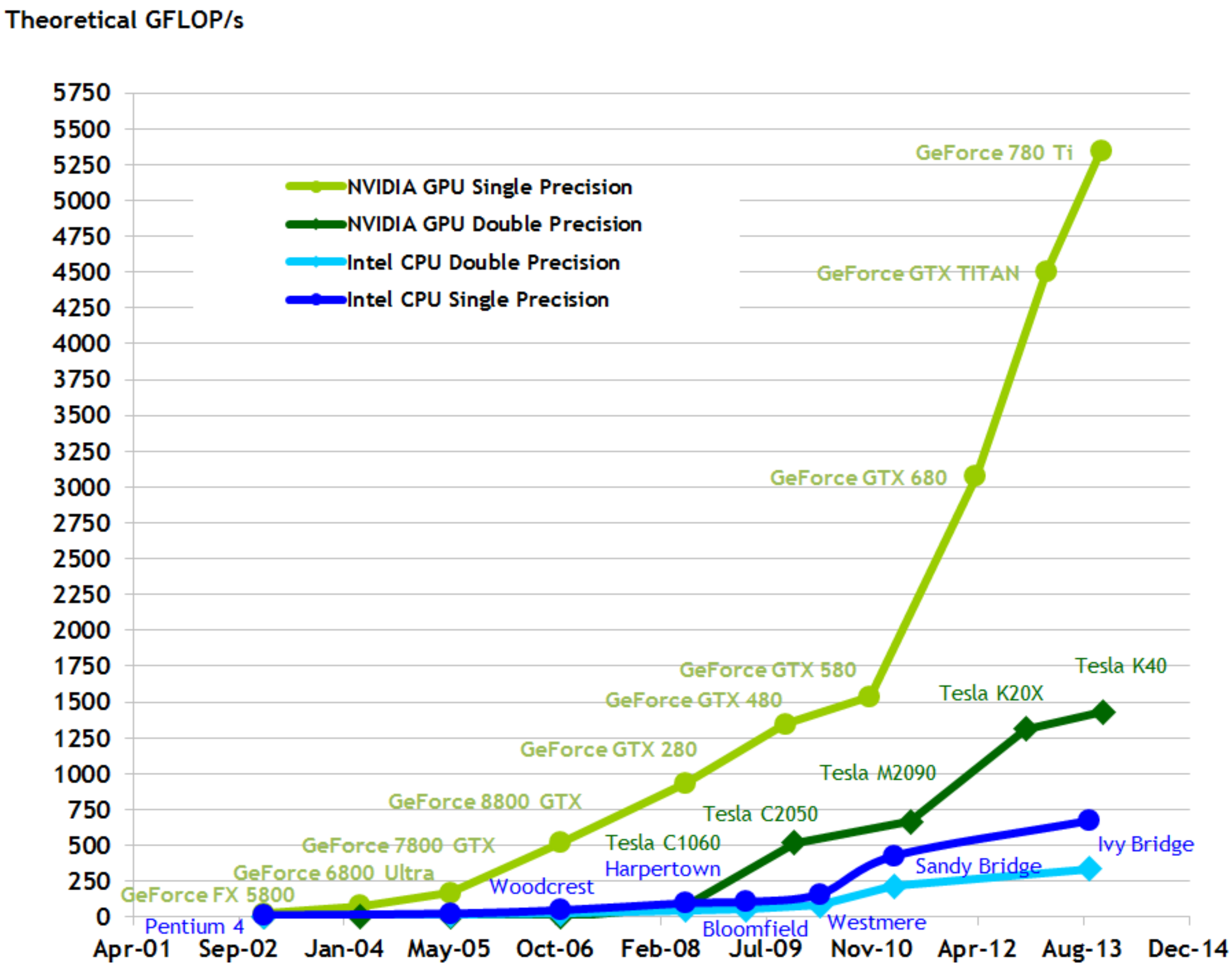
- **HPC trends**
- **3 Challenges**
- **Our approach**
- **Evaluation**

Trend 1: Increasing Heterogeneity

EOL for Von Neumann Frequency Scaling



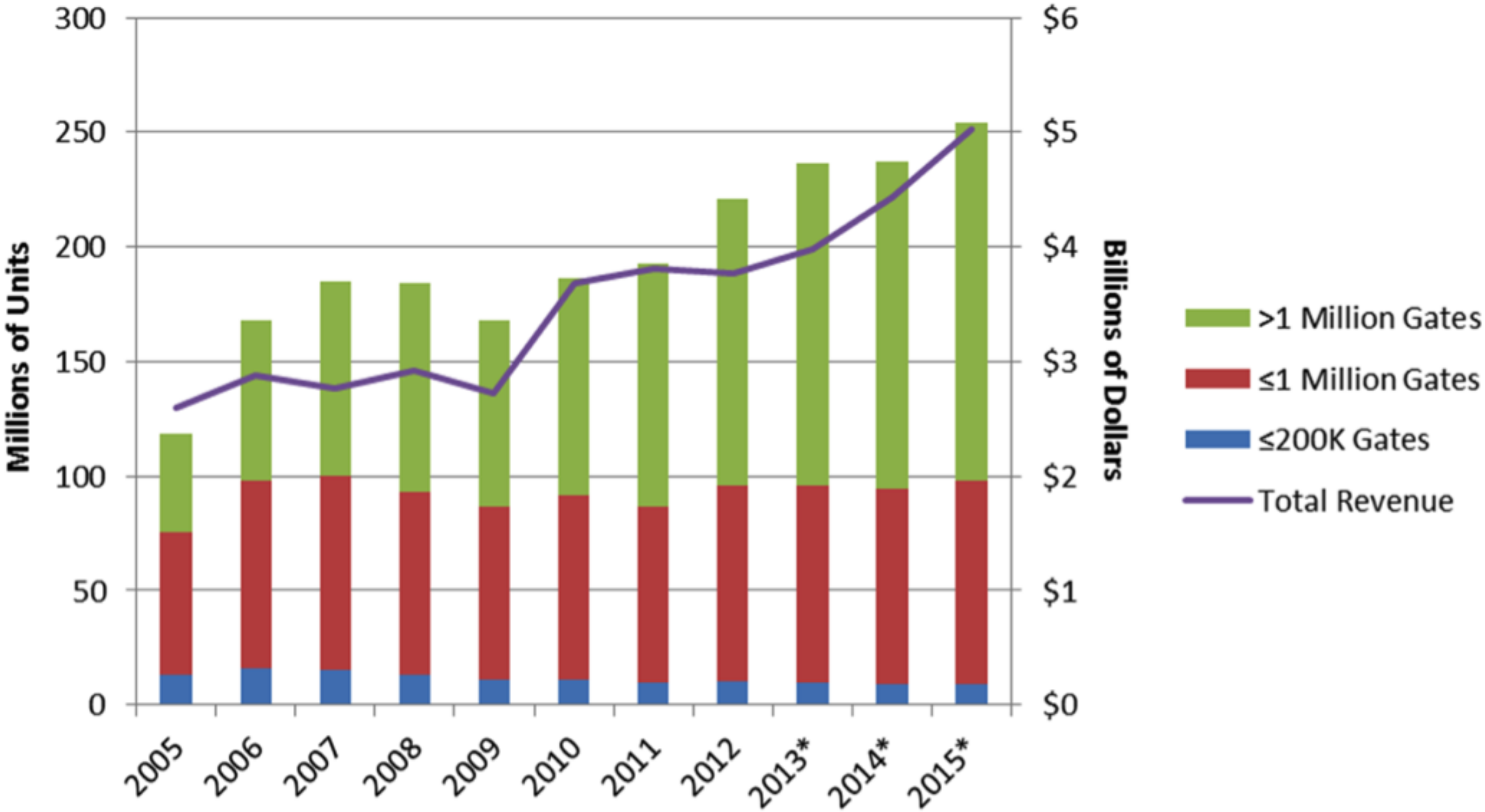
Rise of Alternatives



Source: NVIDIA

Multicore CPU and GPU Performance Growth

Rise of Alternatives



Source: Semico Research Corp

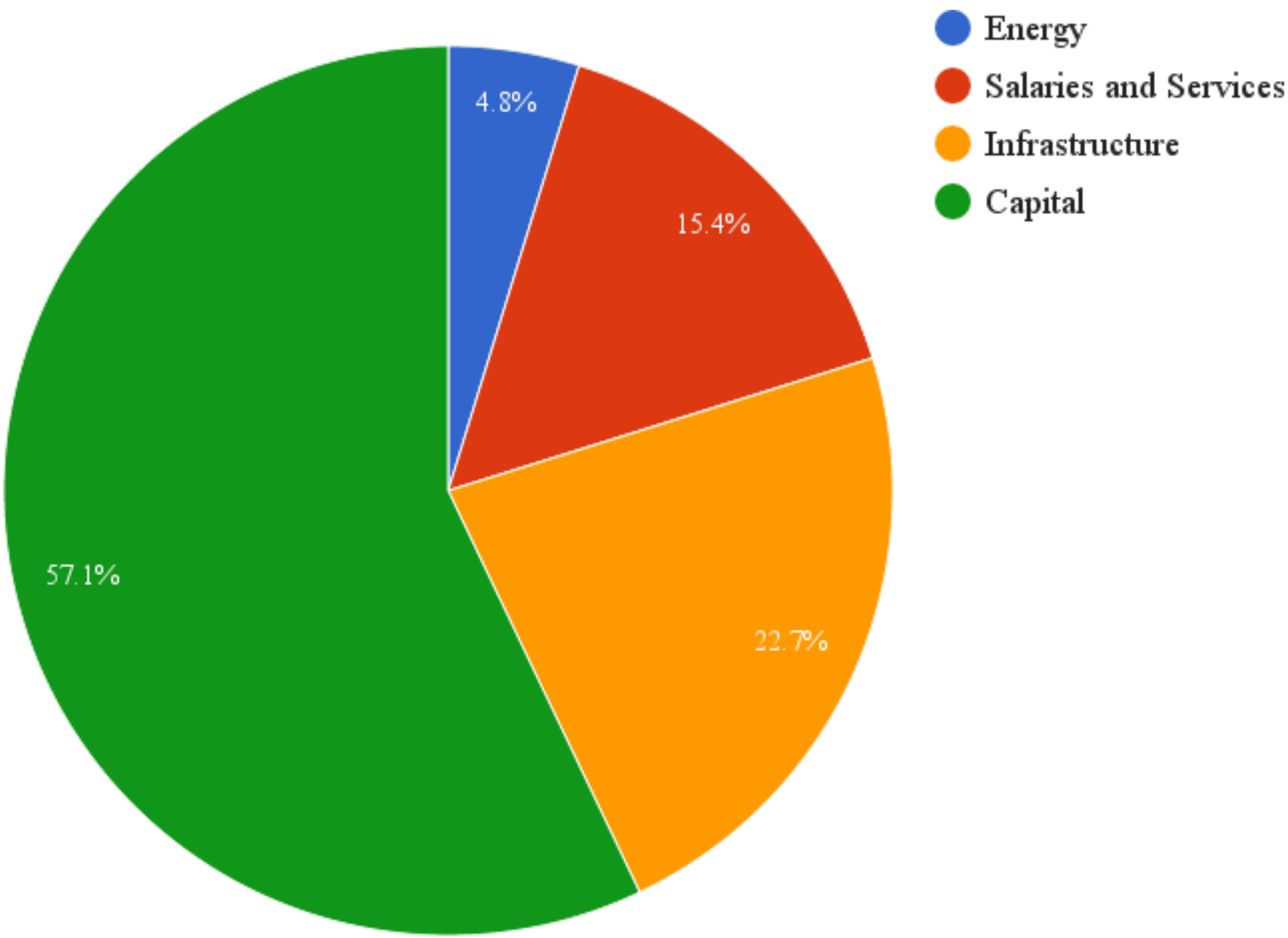
FPGA Market Evolution

Trend 2: Infrastructure-as-a-Service

Providers	**Type**	**Theoretical Peak Performance (*TFLOPS*)**	**Rate (*$/hour*)**
Google Compute Engine	MCPU	~1.6	1.280
Microsoft Azure	MCPU	~1.2	9.65
Amazon Compute Engine	MCPU	1.8	1.856
Amazon Compute Engine	GPU	9.16	2.6

IaaS Performance/Cost Breakdown

Where does all the money go?



3 Challenges

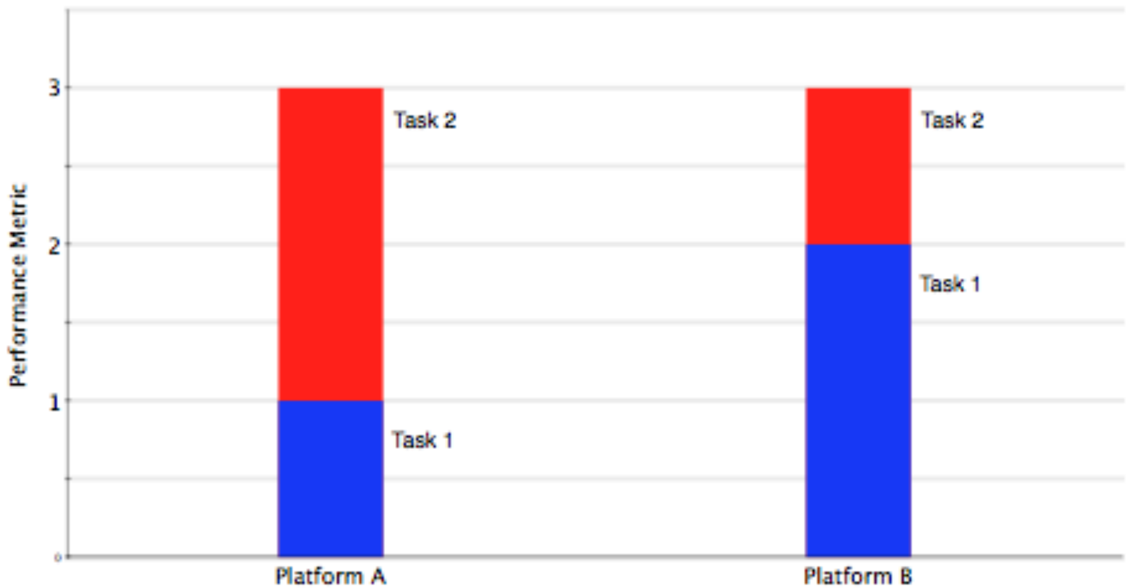
How do I:

1. Execute my tasks on distributed, heterogeneous platforms?

2. Predict the runtime characteristics of my executions?

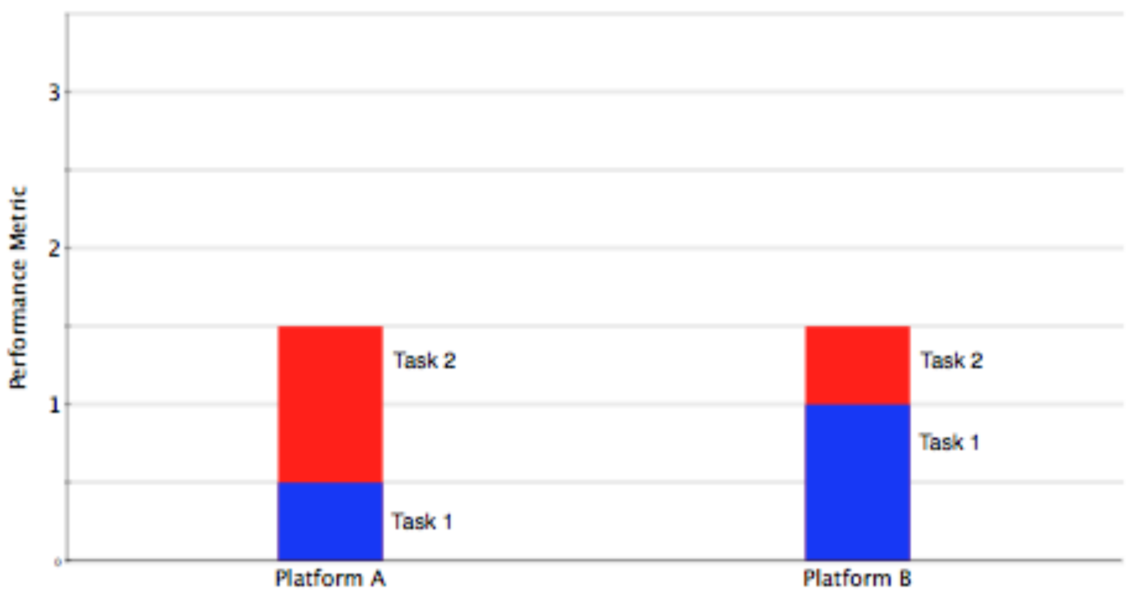
3. Use my resources efficiently?

The Possibility: Superlinear Performance



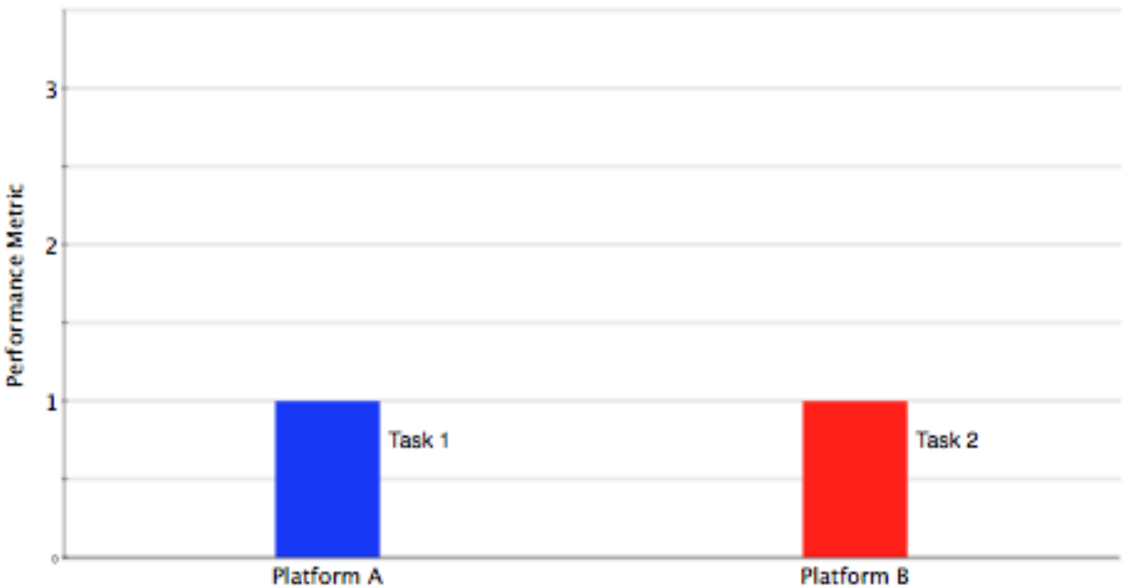
	Task 1	Task 2
Platform A	1.0	1.0
Platform B	1.0	1.0

The Possibility: Superlinear Performance



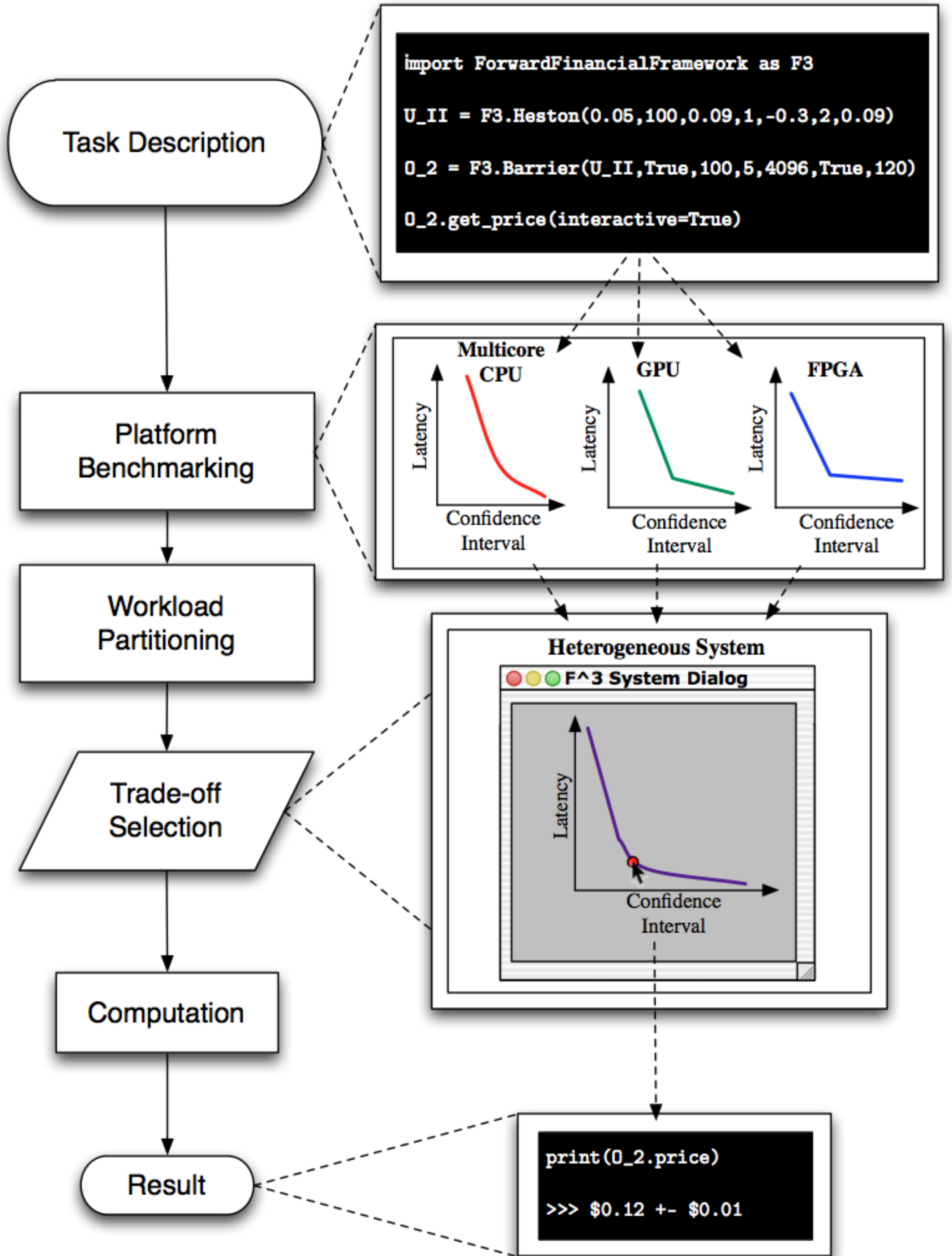
	Task 1	Task 2
Platform A	0.5	0.5
Platform B	0.5	0.5

The Possibility: Superlinear Performance



	Task 1	Task 2
Platform A	1.0	0.0
Platform B	0.0	1.0

Our Approach



Task Description
Platform Benchmarking
Workload Partitioning
Trade-off Selection
Computation
Result
import ForwardFinancialFramework as F3
U_II = F3.Heston(0.05,100,0.09,1,-0.3,2,0.09)
O_2 = F3.Barrier(U_II,True,100,5,4096,True,120)
O_2.get_price(interactive=True)
Multicore CPU
GPU
FPGA
Latency
Confidence Interval
Heterogeneous System
F^3 System Dialog
print(O_2.price)
>>> $0.12 +- $0.01

Application Domain

- Natural grouping of computational operations and types

- Manifest as Domain Specific Languages and Application Libraries

- Result from empirical software engineering show that typically 10-15 high level operations usually dominate utilisation

3 Solutions

1. **Portable Performance**: Exploit domain power law distributions

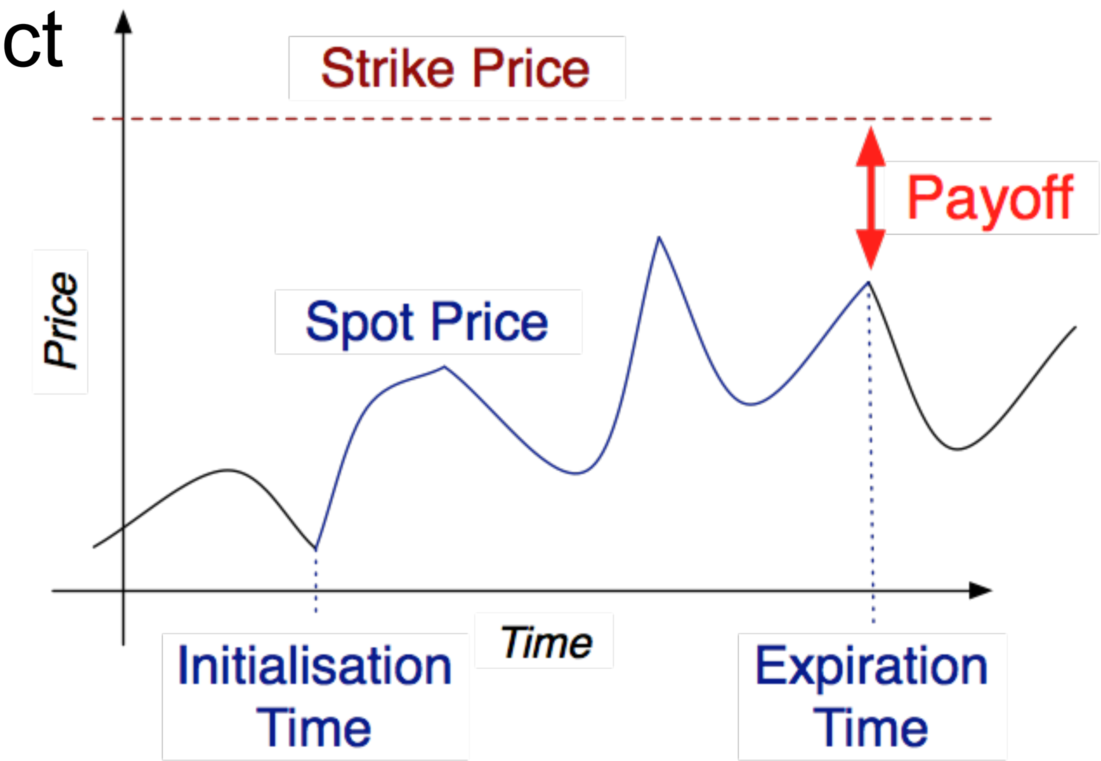
2. **Metric Modelling**: Use domain knowledge to identify and populate models in advance

3. **Efficient Partitioning:** Use metric models and formal optimisation to balance user objectives

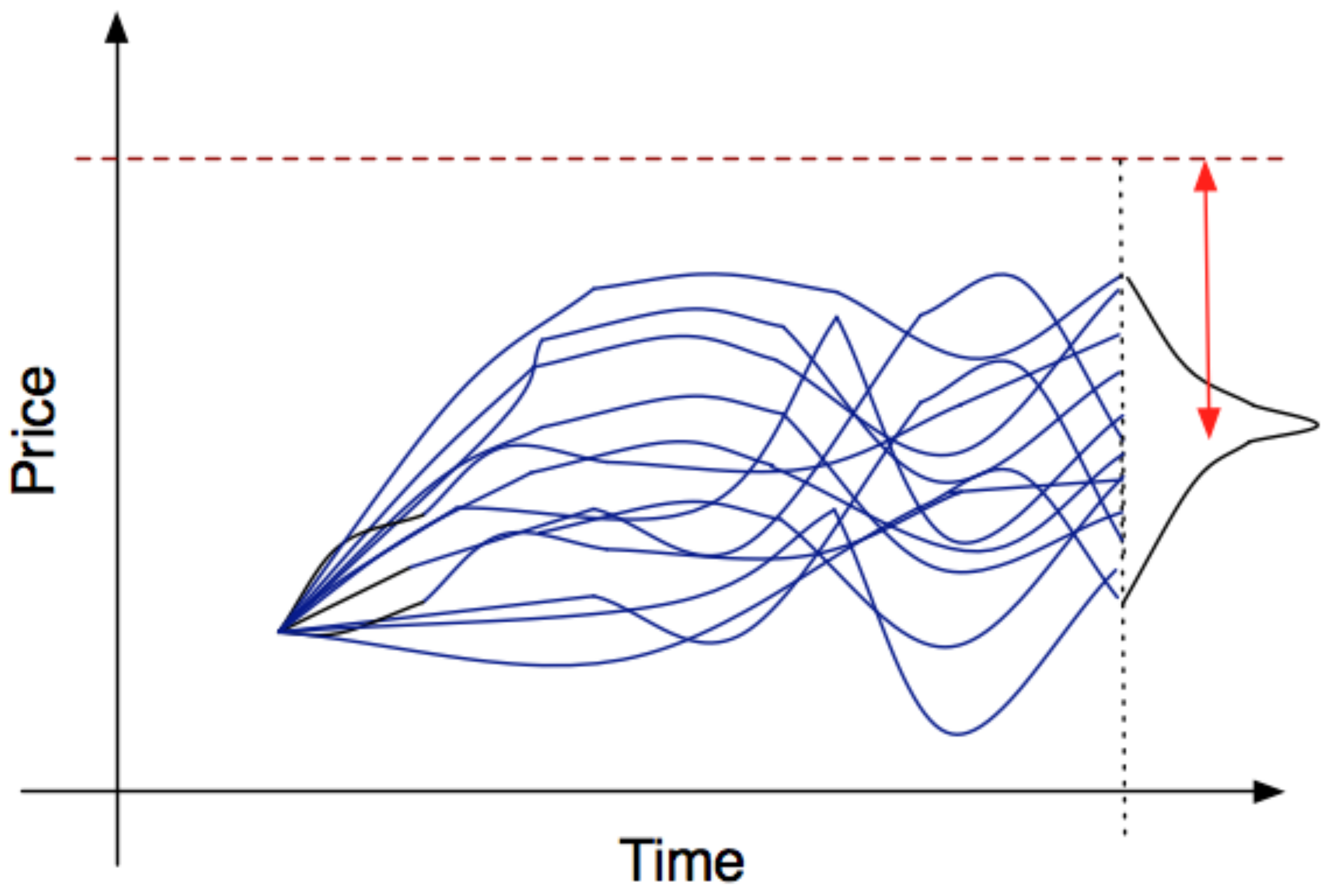
Evaluation

Our Domain:
Forward Looking Option Pricing

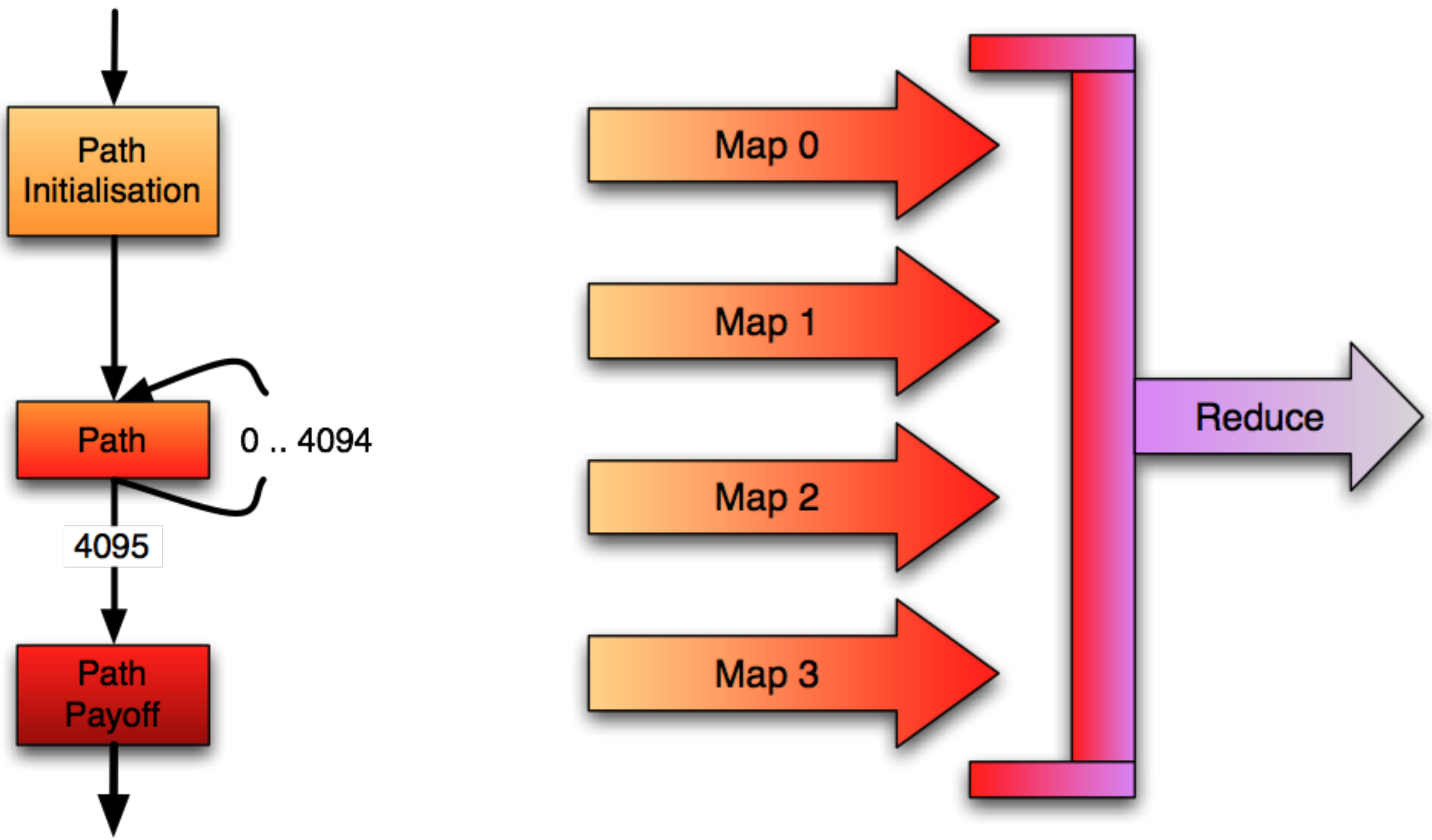
- Finding the value of a derivative contract
- Two Types: Underlyings and Derivatives
- One Operation: Pricing



Monte Carlo Option Pricing



Monte Carlo Pricing as Map Reduce



Our Application Framework: Forward Financial Framework (F^3)

- Python-based Application Framework
- Backends - open standards & platform tools:
  - POSIX + GCC
  - OpenCL + Vendor tools
  - OpenSPL + Maxeler

```
import ForwardFinancialFramework as F3

HU_II = F3.Underlying.Heston(0.05,100,0.09,1,-0.3,2,0.09)
EO_1 = F3.Option.European(HU_II,True,100,5)

P_A = F3.Portfolio([EO_1])
P_A.get_prices()

print EO_1.price
```

Experimental Tasks

- Portfolio Evaluation:
  - 35 x Black-Scholes Barrier and Asian Options
  - 93 x Heston Model European, Barrier and Asian Option
- Scale:
  - 35 MFLOP per simulation of all options
  - 10M - 100M simulations required
  - PetaFLOP scale computation

Experimental Platforms - CPUs

- Tool: GCC 4.8 using POSIX threads

- Local:
  - Desktop - Intel Core i7-2600 (7 threads)
  - Local Server - AMD Opteron 6272 (64 threads)
  - Local Pi - ARM 11 (1 thread)

- Remote:
  - Remote Server - Intel Xeon E5-2680 (32 threads)
  - AWS EC1 & WC1 - Intel Xeon E5-2680 (16 threads)
  - AWS EC2 & WC2 - Intel Xeon E5-2670 (7 threads)

Experimental Platforms - GPUs

- Tool: NVIDIA, Intel and AMD SDKs for OpenCL

- Local:
  - Local GPU 1 - AMD Firepro W5000
  - Local GPU 2 - NVIDIA Quadro K4000

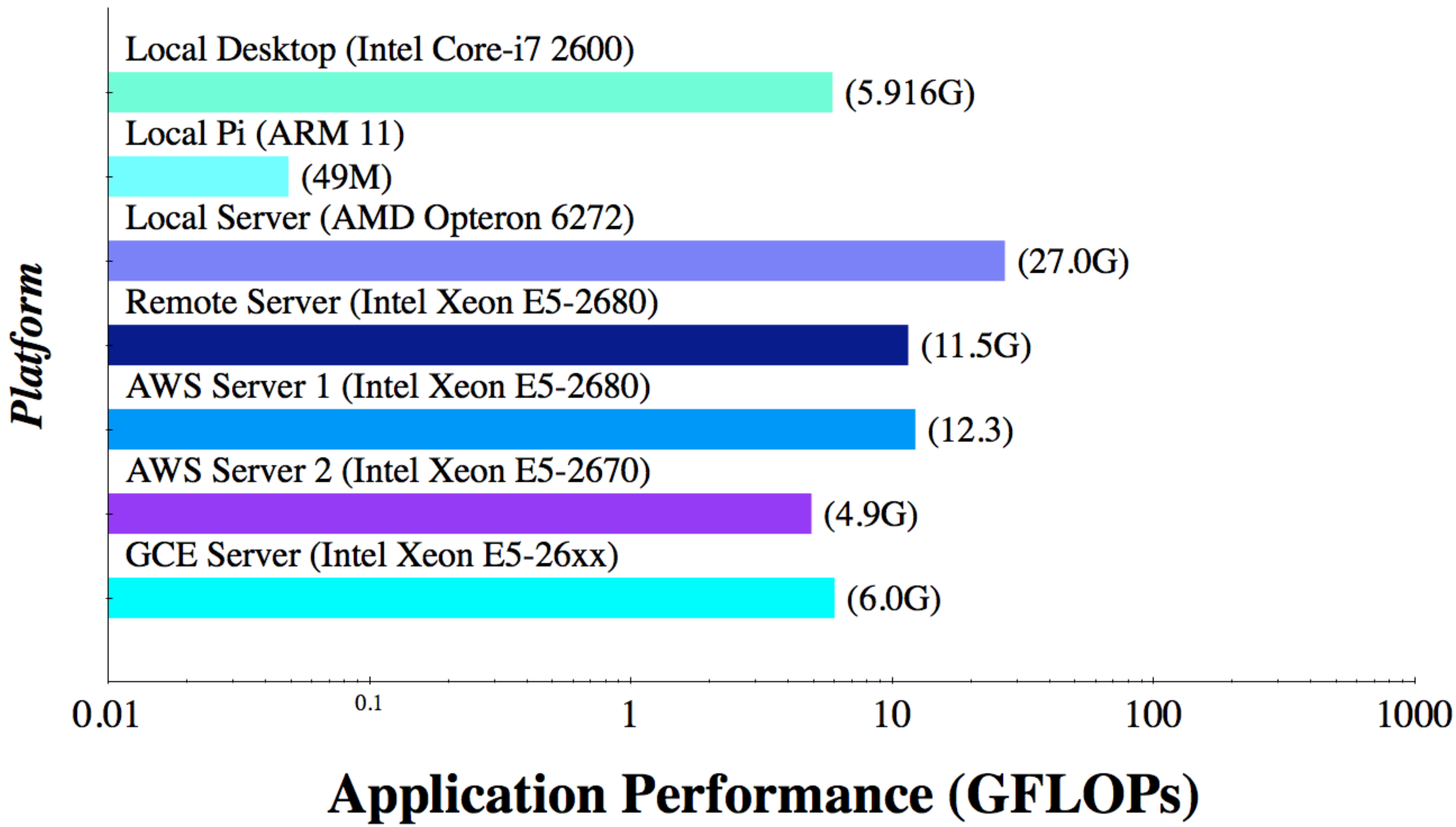
- Remote:
  - Remote Phi - Intel Xeon Phi 3120P
  - AWS GPU EC and GPU WC - NVIDIA Grid GK104

Experimental Platforms - FPGAs

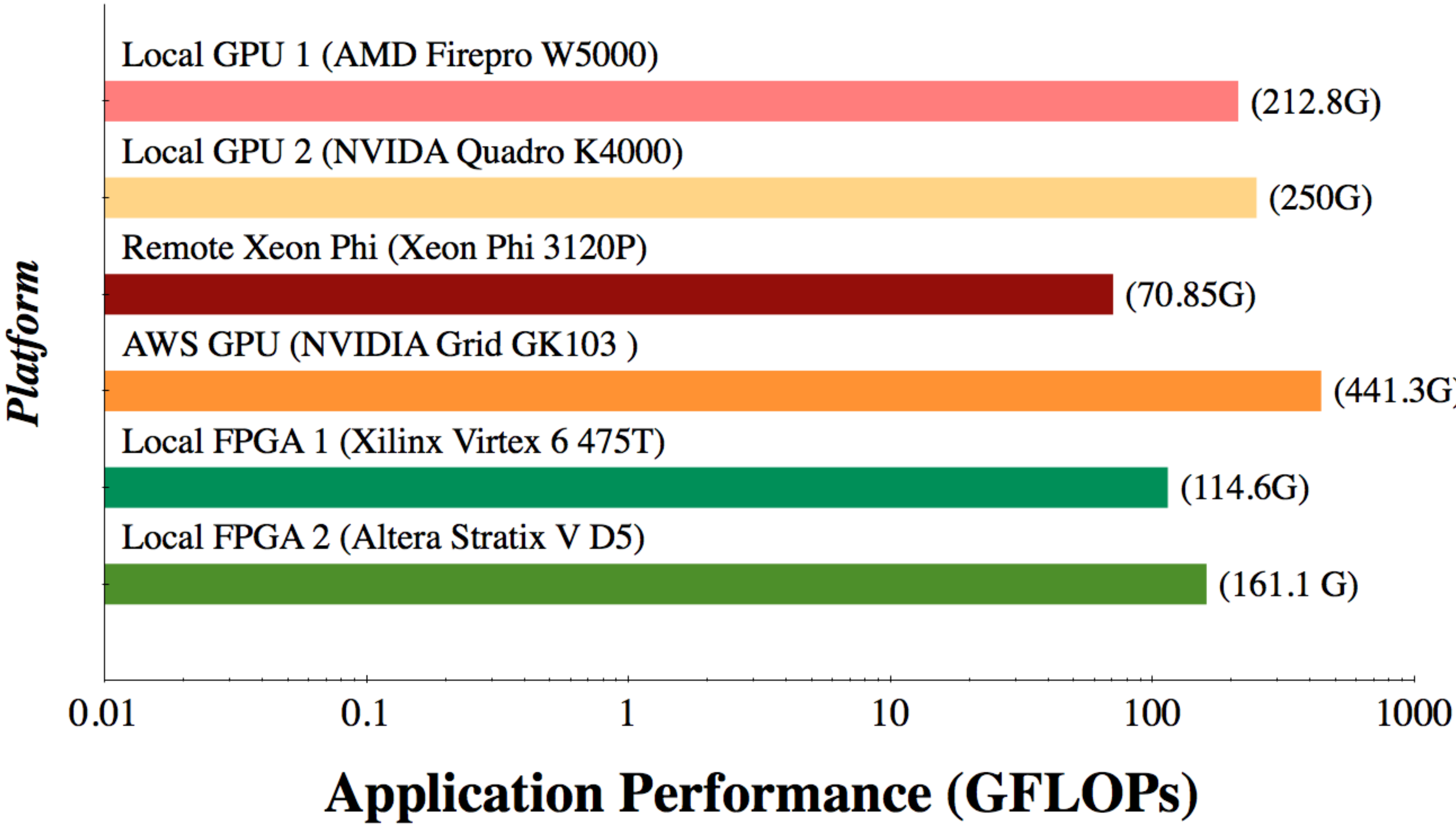
- Tool: Maxeler Maxcompiler and Altera OpenCL SDK

- Local:
  - Local FPGA 1 - Xilinx Virtex 6 475T
  - Local FPGA 2 - Altera Stratix V D5

Portable Performance



Portable Performance



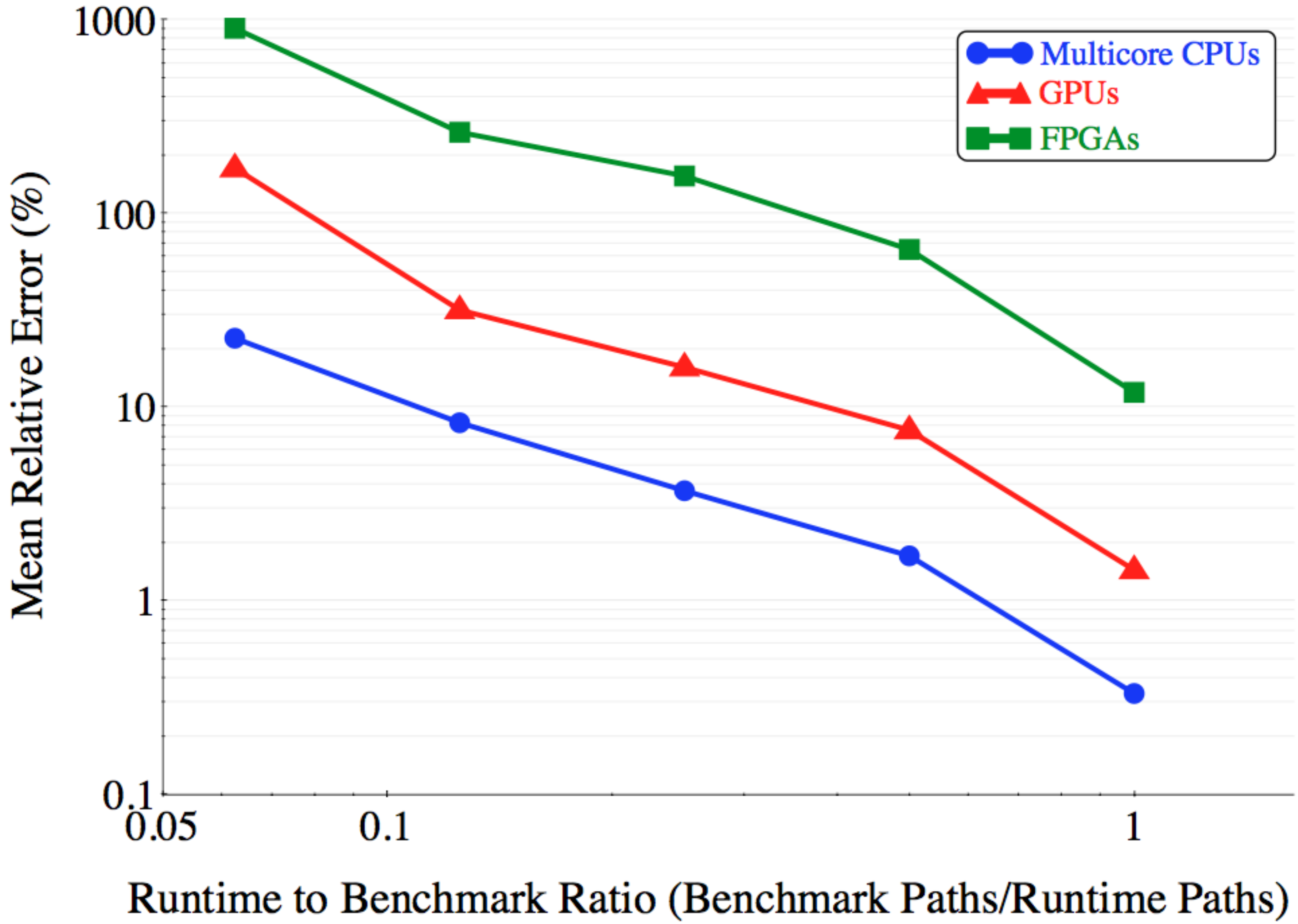
Metric Modeling

- Domain Metrics:
  - Makespan (in seconds)
  - Accuracy (size of 95% confidence interval)

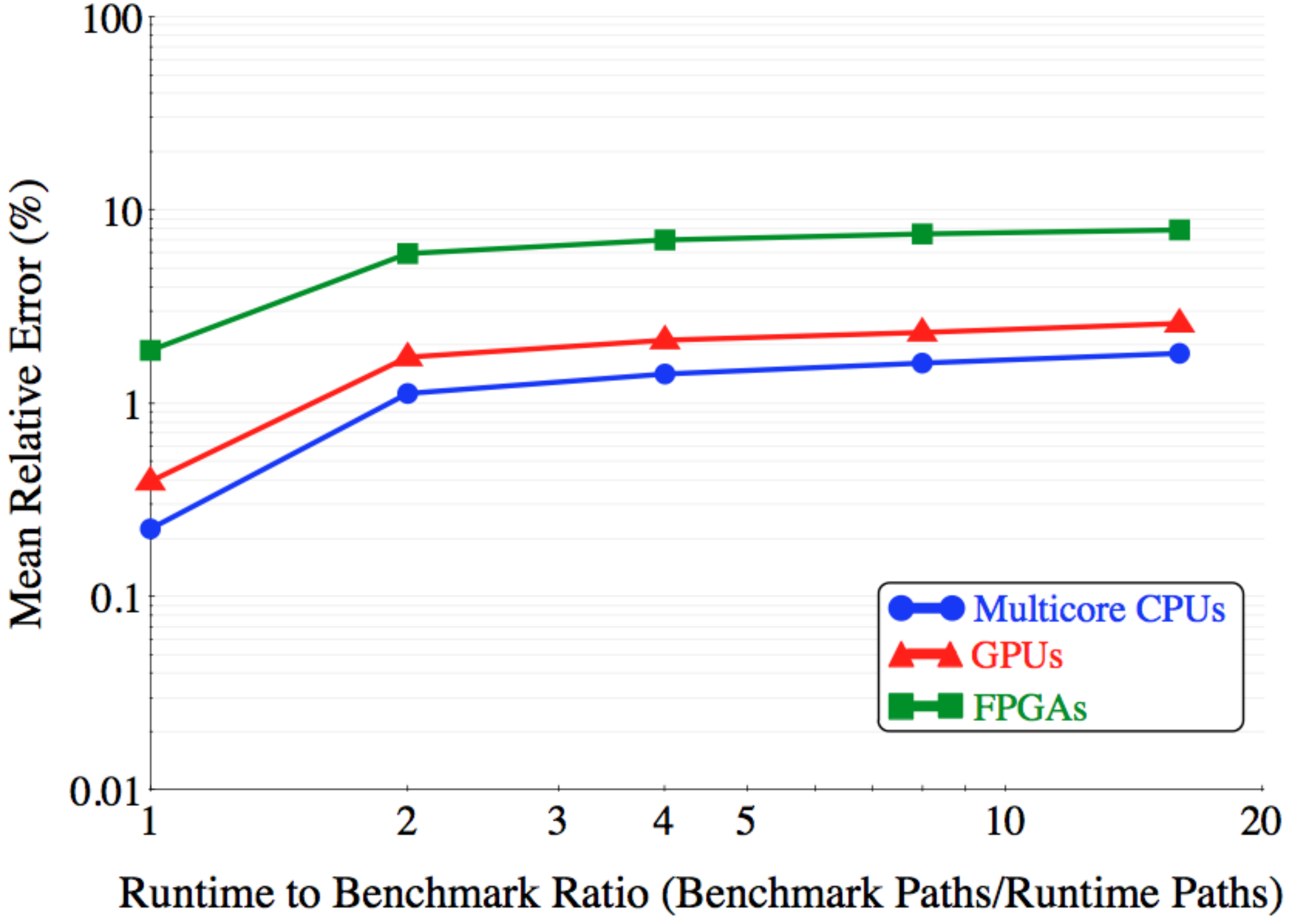
- Latency Model: $L(n) = \beta n + \gamma$

- Accuracy Model: $C(n) = \frac{\alpha}{\sqrt{n}}$

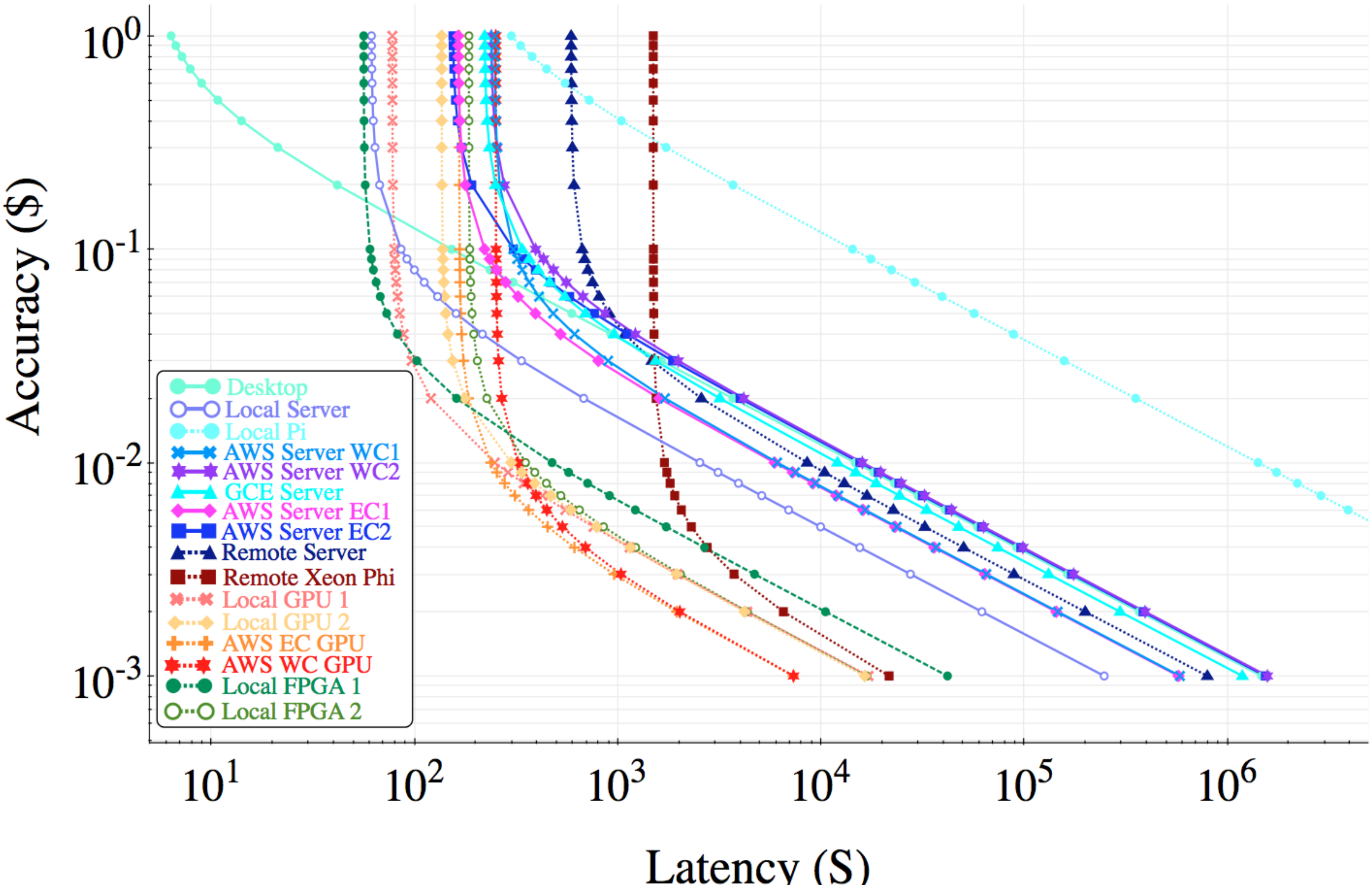
Metric Modeling



Metric Modeling



Metric Modeling



Efficient Partitioning

- Achieve superlinear performance scaling

- Vary allocation to explore design space

- Three approaches:
  - Heuristic
  - Machine Learning-based
  - Formal Mixed Integer Linear Programming

Efficient Partitioning

Metric that we care about

$$\underset{\boldsymbol{A}\in\{0,1\}^{\mu\times\tau}}{\text{optimise}} \quad \boldsymbol{G_k}(\boldsymbol{A}, \boldsymbol{P_p}) \quad \boldsymbol{P_p} \in \mathbb{R}^p$$

$$\text{subject to} \quad \sum_{i=1}^{\mu} A_{i,j} = 1 \quad j = 1, 2, \ldots, \tau$$

Efficient Partitioning

$$\underset{\boldsymbol{A} \in \mathbb{R}_+^{\mu \times \tau}}{\text{minimise}} \quad \boldsymbol{G_L}(\boldsymbol{A}, \boldsymbol{C}) \quad \boldsymbol{C} \in \mathbb{R}_+^{\tau}$$

$$\text{subject to} \quad \sum_{i=1}^{\mu} A_{i,j} = 1 \quad j = 1, 2, \ldots, \tau$$

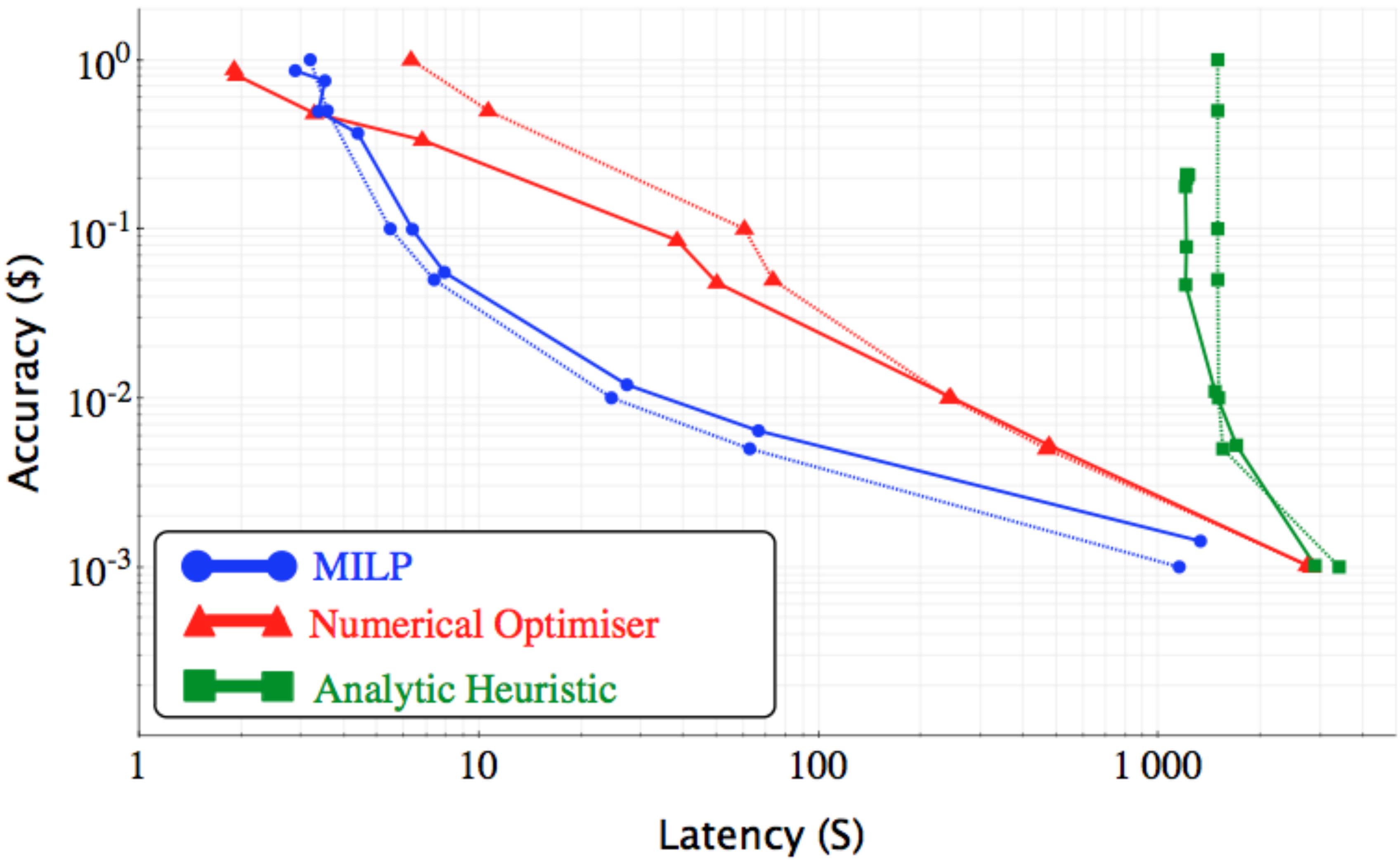
where:

$$\boldsymbol{G_L}(\boldsymbol{A}, \boldsymbol{C}) = \max(\boldsymbol{F_L}(\boldsymbol{A}, \boldsymbol{C}))$$

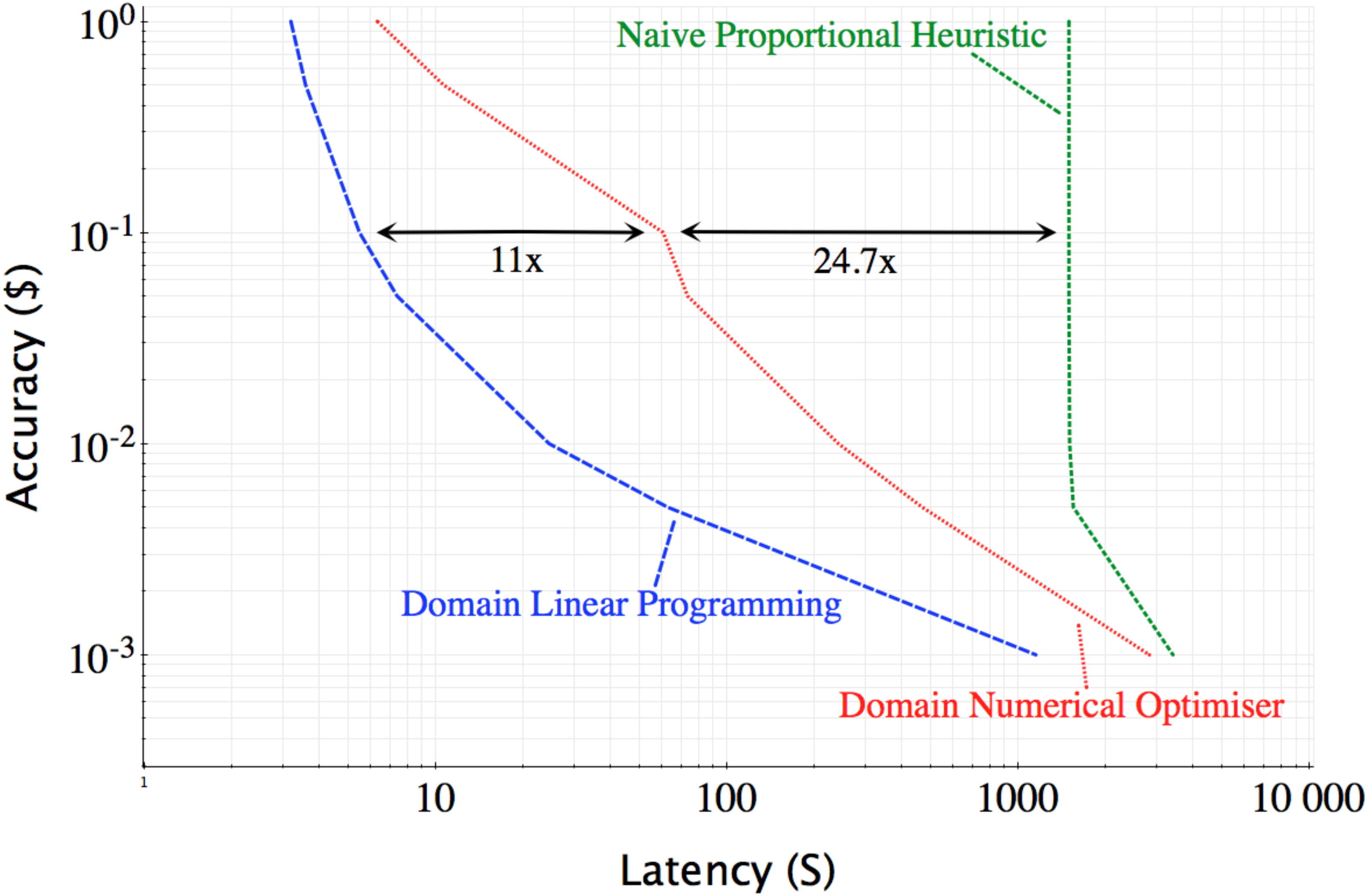
$$\boldsymbol{F_L}(\boldsymbol{A}, \boldsymbol{C}) = (\boldsymbol{\delta} : \boldsymbol{C}^2 \circ \boldsymbol{A} + \boldsymbol{\gamma} \circ \lceil \boldsymbol{A} \rceil) \cdot \mathbf{1}$$

$$\boldsymbol{\delta} \in \mathbb{R}_+^{\mu \times \tau}, \boldsymbol{\gamma} \in \mathbb{R}_+^{\mu \times \tau}$$

Efficient Partitioning



Efficient Partitioning



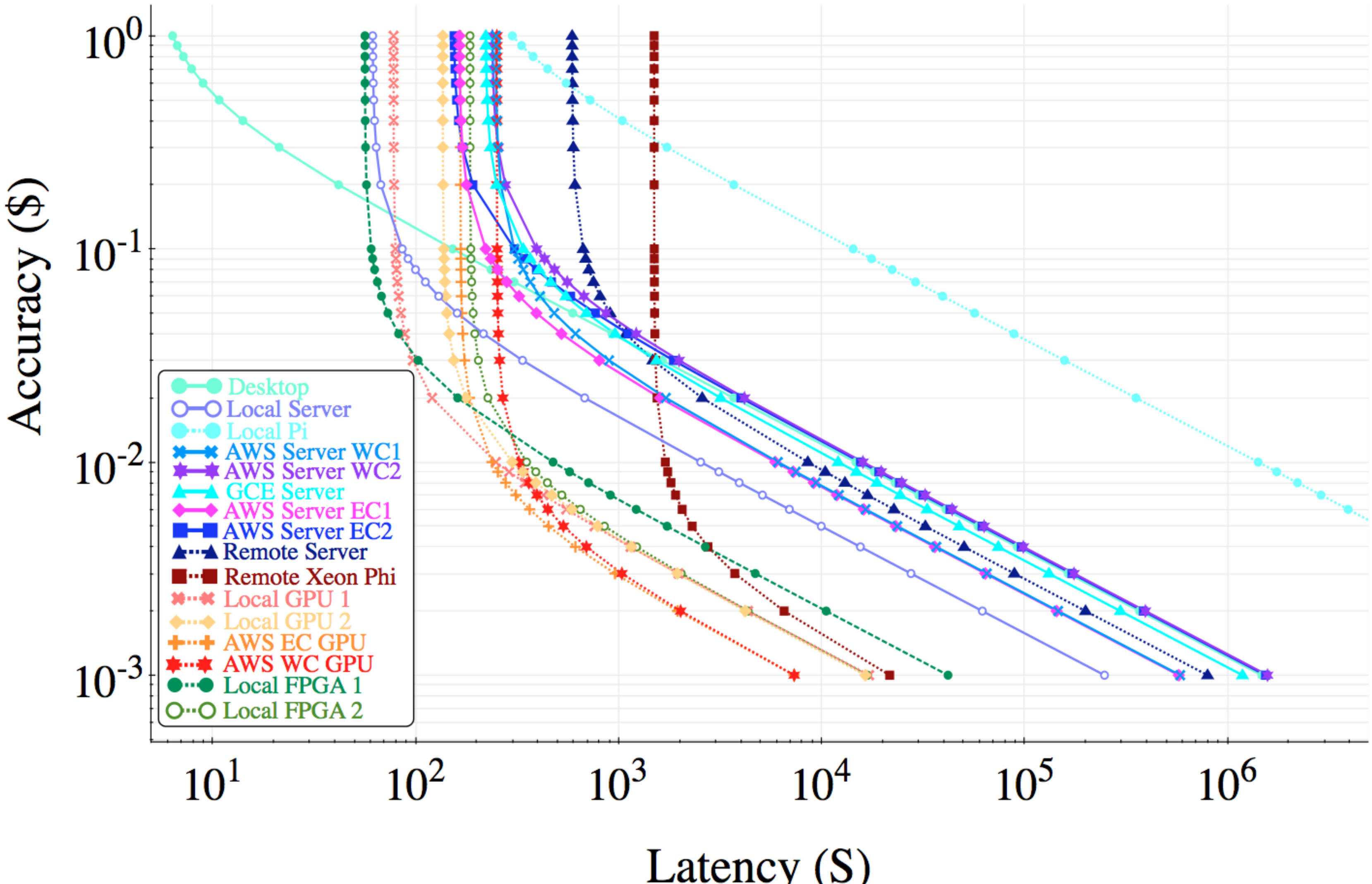
- **HPC trends and Challenges**

- **Our domain specific approach:**
  - **Explicit Parallelism**
  - **Metric Models**
  - **Formal Optimisation**

- **Evaluation**

Thanks!

Metric Modeling



Efficient Partitioning