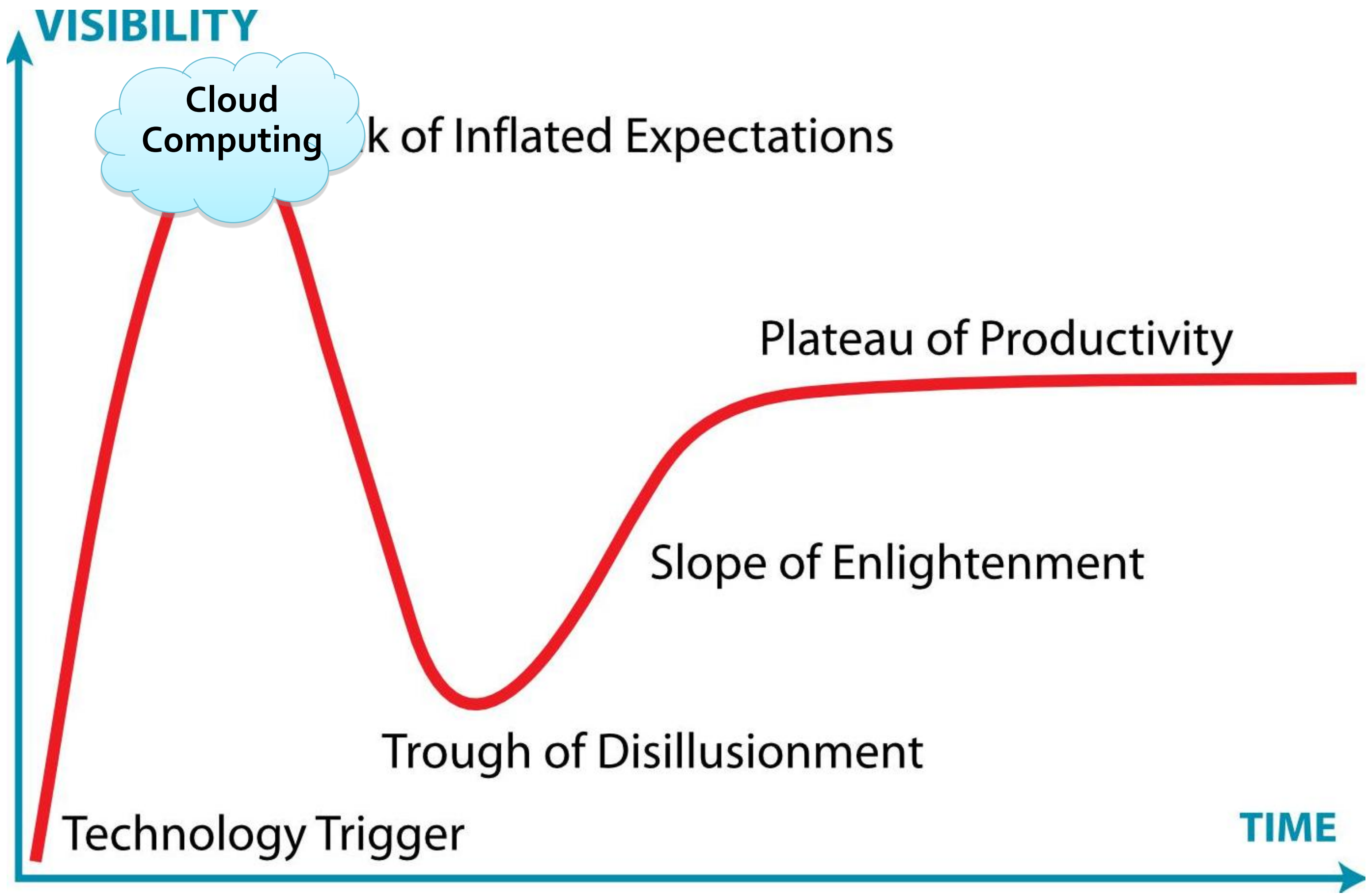
ORACLE®

The Coming Age of Specialized (Cloud) Computing Systems
IEEE Technology Time Machine – Dresden 2012

Eric Sedlar
Technical Director, Oracle Labs

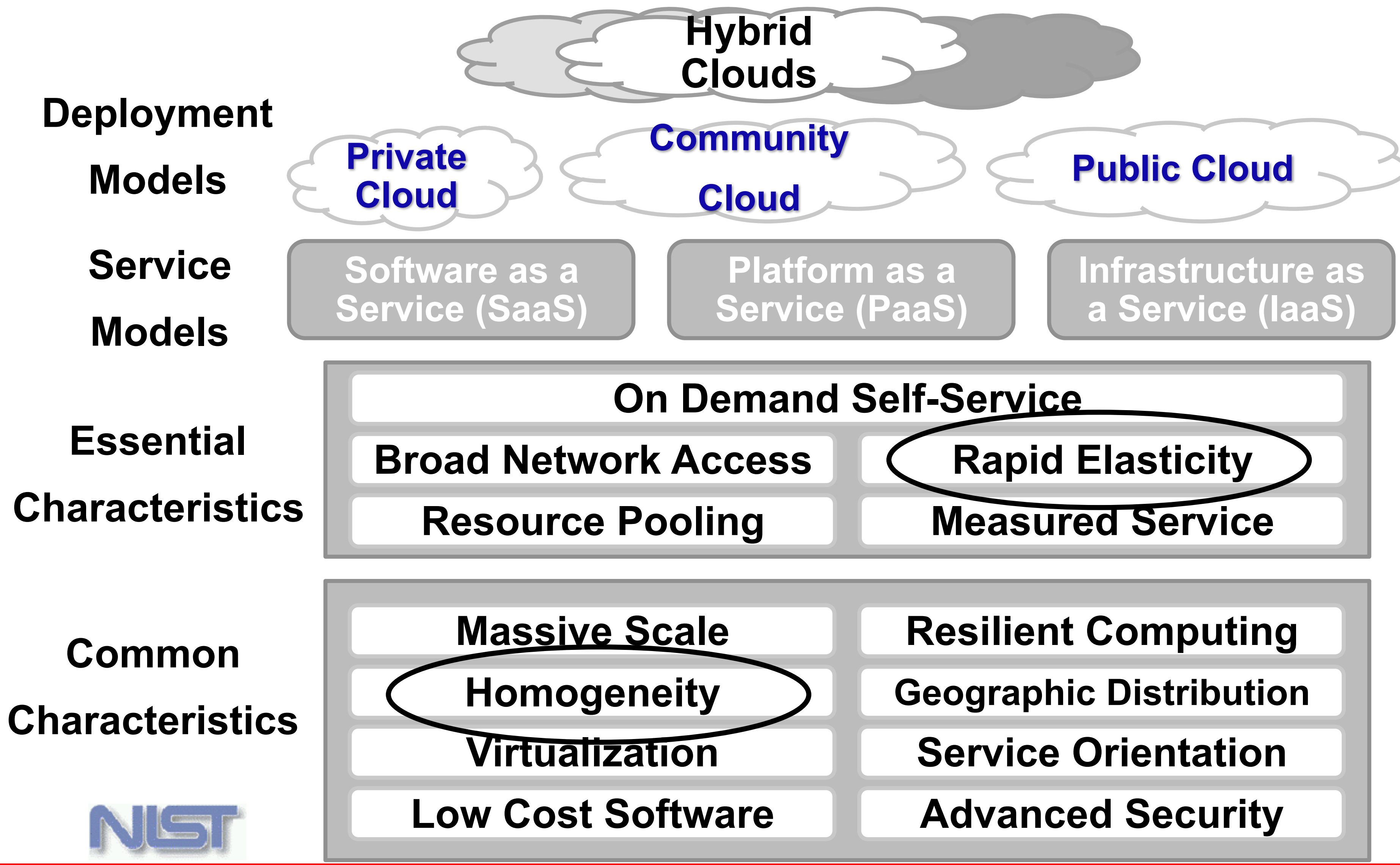
Cloud Computing: An Economics-Driven Wave



* From http://en.wikipedia.org/wiki/Hype_cycle

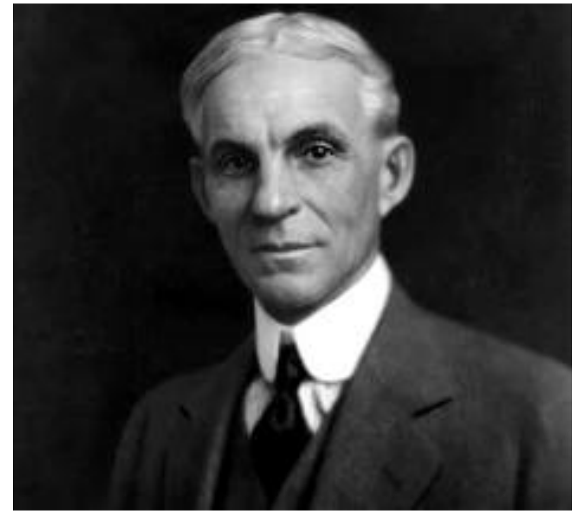
ORACLE

NIST Cloud Definition Framework



ORACLE

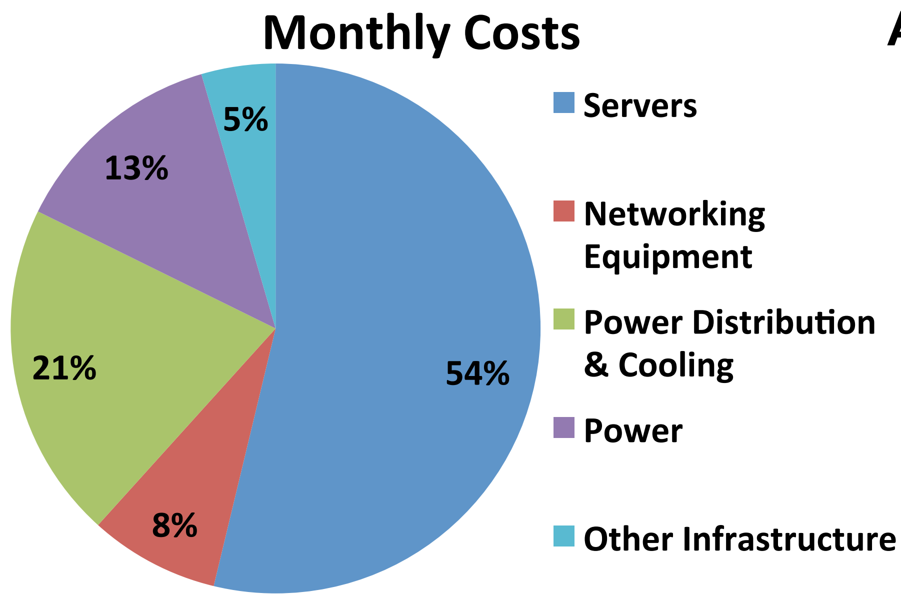
Homogeneity to Amortize Fixed Costs & Achieve Economies of Scale



- "*Any customer can have a car painted any color that he wants so long as it is black*" – Henry Ford, 1909
- Amazon believes there is around a 7x cost savings for hardware & administrative costs at Internet scale vs. Enterprise Scale
  - Fewer economies of scale for power-related costs
- Generic cloud apps: Facebook, Twitter, Amazon, Gmail
- Custom cloud apps most frequently seen in the App Store
- Nearly 100% of Oracle On-Demand Enterprise Applications have requirements for customizations: SQL tables, Java classes, etc.

ORACLE

Economic Driving Factors in the Cloud—Amazon



3yr server, 4yr net gear, & 10 yr infrastructure amortization

Assumptions:

Facility: ~$88M for 8MW facility
Servers: Roughly 46k @ $1.45k each
Server power draw at 30% load: 80%
Commercial Power: ~$0.07/kWhr

Conclusion: 34% of costs related to power (trending upwards)

- Eric's Comments:
  - Cited cost of software development is zero (equal to marginal cost)
    - 100 software developers @ $250k/yr = $25m/yr → not negligible even at cloud scale
  - Don't build your data center in Germany @ 36¢/ KWhr or Denmark @ 40 ¢ / kWhr
  - In many cases power & fixed asset sizing is a hard limit
  - Benefits of cloud scale come from amortization HW & admin scale

ORACLE

Elasticity vs. Heterogeneity

- The more heterogeneous infrastructure is required to run applications, the less elastic the cloud may be
  - If switching a server/core/cluster from one application to another has high startup costs, response time will suffer
  - Application-specific hardware may not be fungible: what percentage of the time is the GPU on your laptop used?
- Elasticity has the same kinds of benefits and limits as asset diversification does on a financial portfolilo
  - There still may be many “black swans” or correlated events that require significant over-provisioning
- As energy costs become more critical, over-provisioning HW means less as long as it is power-gated

ORACLE

Why does Cloud Computing Require Smarter Application Developers?

- Google, Facebook & Amazon hire lots of the worlds smartest engineers to build many applications that are fairly straightforward outside of their scalability requirements
- Application development at Internet scale requires lots of deep systems thinking about programming models, data movement, and transactional consistency
  - All of this gets in the way of doing what the customer wants
- Most custom apps (Mobile or Enterprise) aren't running at Internet scale

My Argument in a Nutshell

- Cloud computing driven by economies of scale, but…
  - the economics assumes SW cost = 0 while pushing complexity into applications to get the necessary scale!
    - only works for generic apps
  - Elasticity helps drive up utilization & drive down HW costs
    - Elasticity may drive utilization up 2X, reducing HW cost by 2
      - But specialization can get you an order of magnitude in efficiency
    - Since HW costs & power costs are similar, elasticity is secondary
  - However, HW specialization also drives up SW costs
- Better software abstractions are needed to address productivity costs of cloud scale (parallelism) and specialized HW

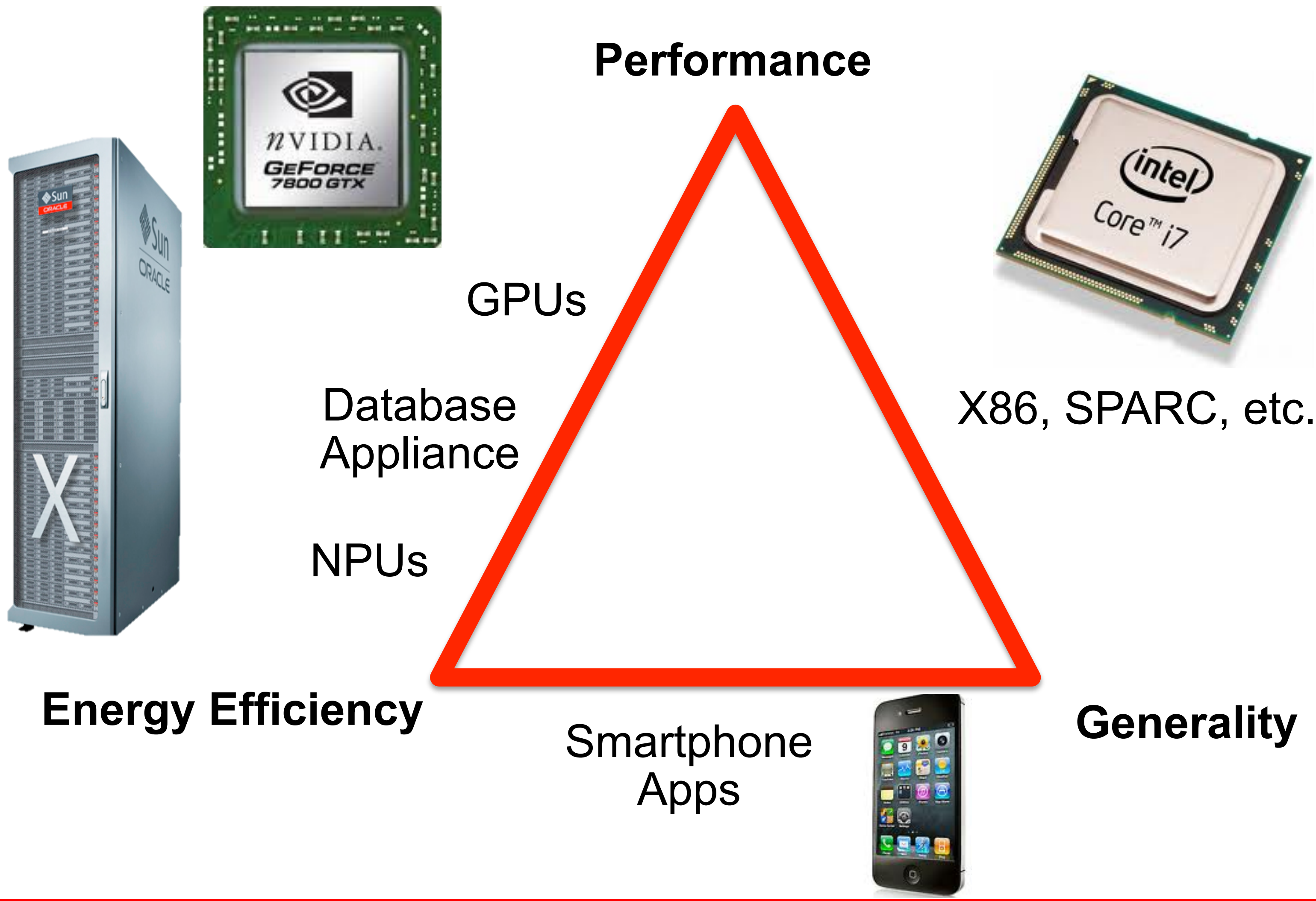
ORACLE

The Great Whales of Computing Research: Power Efficiency & Parallelism at Scale

- Power considerations force more parallelism
- Programming parallel systems at scale is too hard
- Homogeneity & generality are the core limits

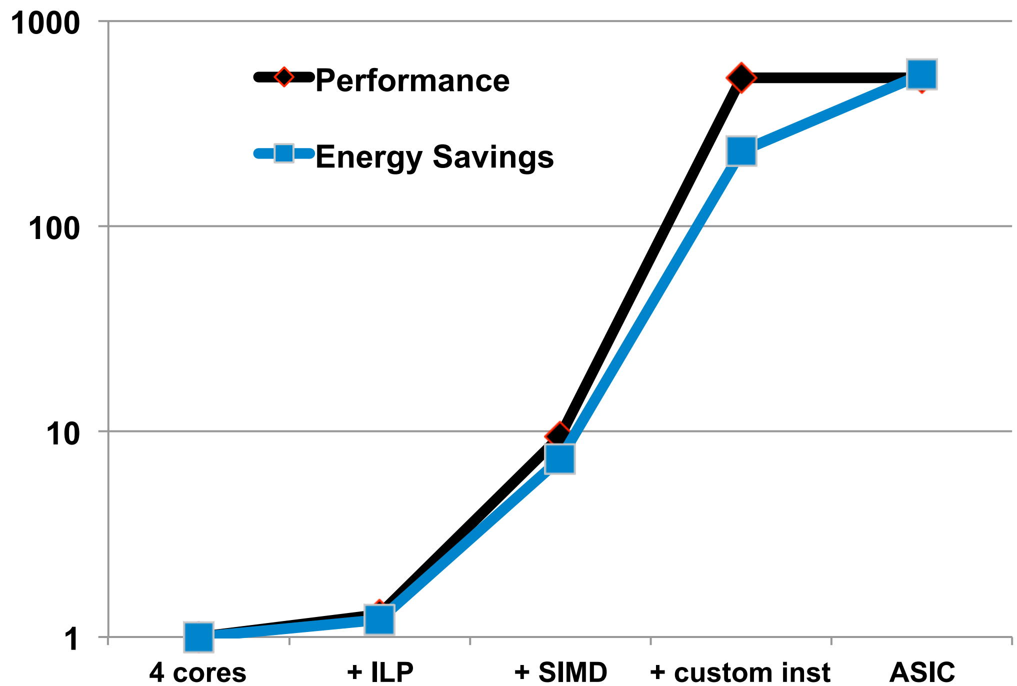
- *Will scalability continue to be a big issue for Internet Applications?*
- *Will Internet usage growth rates drop below Moore's Law?*
  - Internet usage went from 1.1b to 2.2b users in the last 5 years
  - Transistor counts doubling every 2 years (ITRS roadmap to 2026 says we'll slow to doubling every 3 years)

Limiting Generality Enables More Efficient Hardware



Big power savings in the processor with application-specific hardware

- H.264 encode study



Horowitz et al. *Understanding Sources of Inefficiency in General-Purpose Chips* (ISCA 2010)

- Anton molecular dynamics computer

- **400 MHz, 100x power savings**
- **1000x performance improvement**
- **Supercomputing 2009 Best Paper**

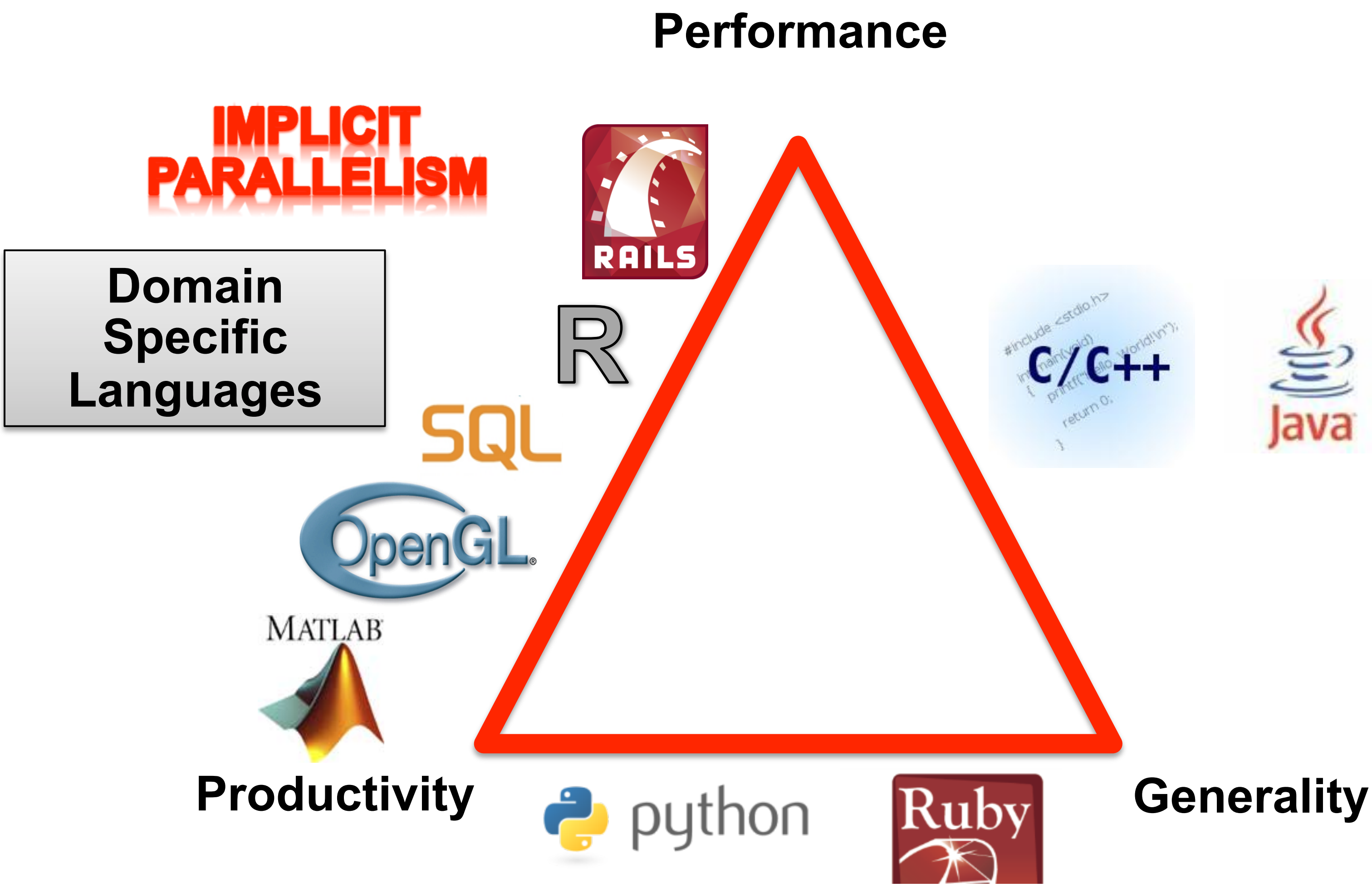
Specialized HW is growing

- Mobile phones:
  - Processing moving from the general purpose ARM processor to DSPs and fixed-function units
- Network processing
  - Vendors like Netlogix and Cavium as well as FPGA vendors like Xilinx are adding more fixed function silicon
- Engineered systems like Exadata can gain 10X in perf.
  - Specialize rack components by pushing scans to the storage nodes
- Even Intel is adding application-specific ISA extensions
- Hardware refresh cycles are less than or equal to the product release cycles for platform software: ~2 years
- Vertical integration is returning
  - IBM & Apple, now Oracle and Google

Challenges for Hardware Specialization

- Difficult to identify legacy software that can be accelerated via hardware: two approaches to this
  - HW/SW co-design
  - Higher-level language abstractions (e.g. DirectX)
- Latency & bandwidth between the specialized computing units and the generalized computer core
  - Overhead from OS & PCIe can dwarf accelerator benefits without large granules of work to offload
- Simulator performance for co-designed software
- Development costs at the hardware level still high
  - VeriLog development & verification costs are high & growing as process technology improves
  - Higher-level HW design (e.g. compiling C to HW) still inefficient

Limiting Generality Can Make For More Productive Software Development



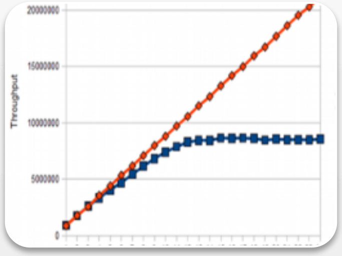
Benefits of Using DSLs for Parallelism

Productivity

- Shield most programmers from the difficulty of parallel programming
- Focus on developing algorithms and applications and not on low level implementation details

Performance

- Match high level domain abstraction to generic parallel execution patterns
- Restrict expressiveness to more easily and fully extract available parallelism
- Use domain knowledge for static/dynamic optimizations

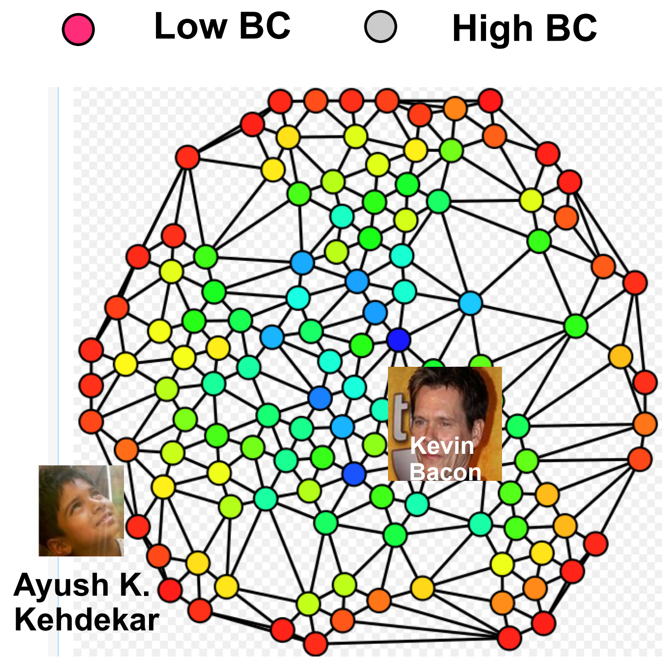


Portability and forward scalability

- DSL & Runtime can be evolved to take advantage of latest hardware features
- Applications remain unchanged
- Allows innovative HW without worrying about application portability

DSLs spreading to more domains

- Application Domains
  - Bioinformatics
  - Physics
  - Financials
  - Network management
- Algorithmic Domains
  - Graph Analysis
  - Statistics
  - Analytics



[Image source; Wikipedia]

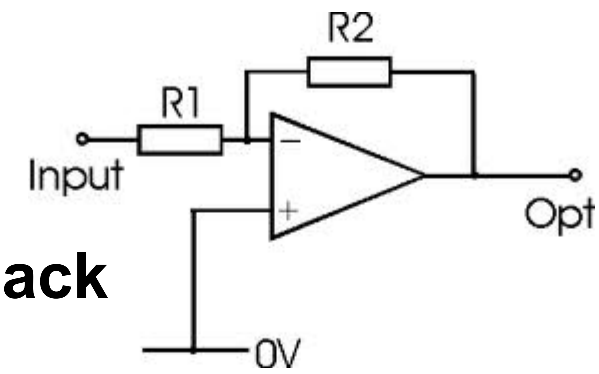
Challenges for Domain Specific Languages

- Issues with DSLs in the past:
  - Extensibility and interoperability—is embedding needed?
  - Long-term viability of the language
    - IP created with languages is typically valuable and long-lived
  - Cost to create the language
  - Availability of tools for language users
  - Learning curve for the language
- Why now?
  - Need for productive ways to exploit parallelism & custom HW
  - Metaprogramming tools to generate compilers & tools can reduce the costs of DSL creation
  - Language tools are improving to get learning curve → library usage
  - The future may see the DSLs agglomerate into more general languages as we understand requirement overlap better

Conclusions & Predictions…

- Power & developer productivity will drive tech directions beyond 2020
  - Not hardware costs or sysadmin costs
  - 20th century has shown that homogeneity is only the first step
- More of us will program in more specialized languages that run on more specialized computers
  - Hardware / software co-design (iPhone, ExaData, …) will grow
  - The number of languages & HW platforms will grow
  - Current technology stacks will obviously continue
- Tweaking the generality tradeoffs
  - Parallel computing at scale currently application-specific
    - Move from application domains to algorithmic domains to increase generality across sets of applications, if not all

Disclaimer: there is another major argument for cloud computing that NIST doesn't cover: continuous feedback



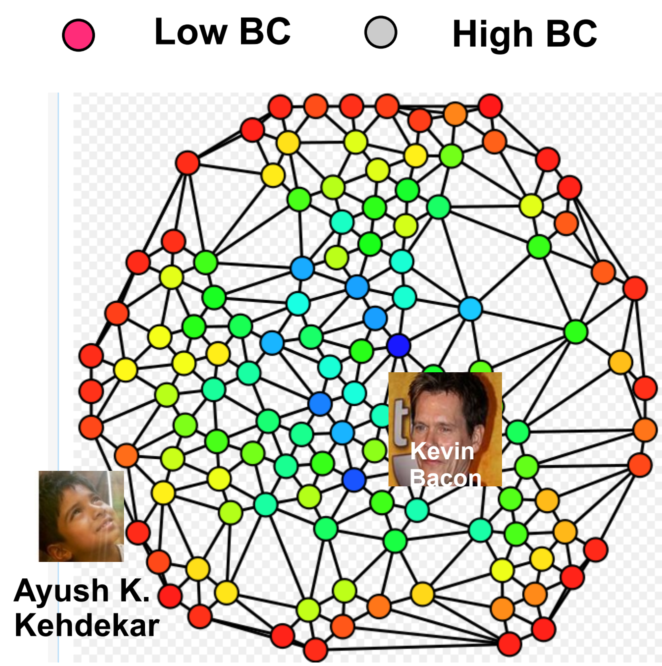
- Many domains need continuous small innovations rather than big-picture architectural changes, e.g. relevance ranking of search results
  - Important when quality of output is not easily quantifiable
- Cloud operators can see apps as they are used
  - Cloud providers can learn from user behavior & their data
  - Straightforward tradeoff of privacy for functionality
- Cloud feedback loops work best when goodness is subjective but requirements are still generalizable across customers

Hardware and Software
Engineered to Work Together

BACKUP SLIDES

Example Graph Algorithm: Betweenness Centrality

- (Node-) Betweenness Centrality
  - A measure that tells how ‘central’ a node is in the graph
  - Math. Definition
    - How many shortest paths between any two nodes goes through this node.



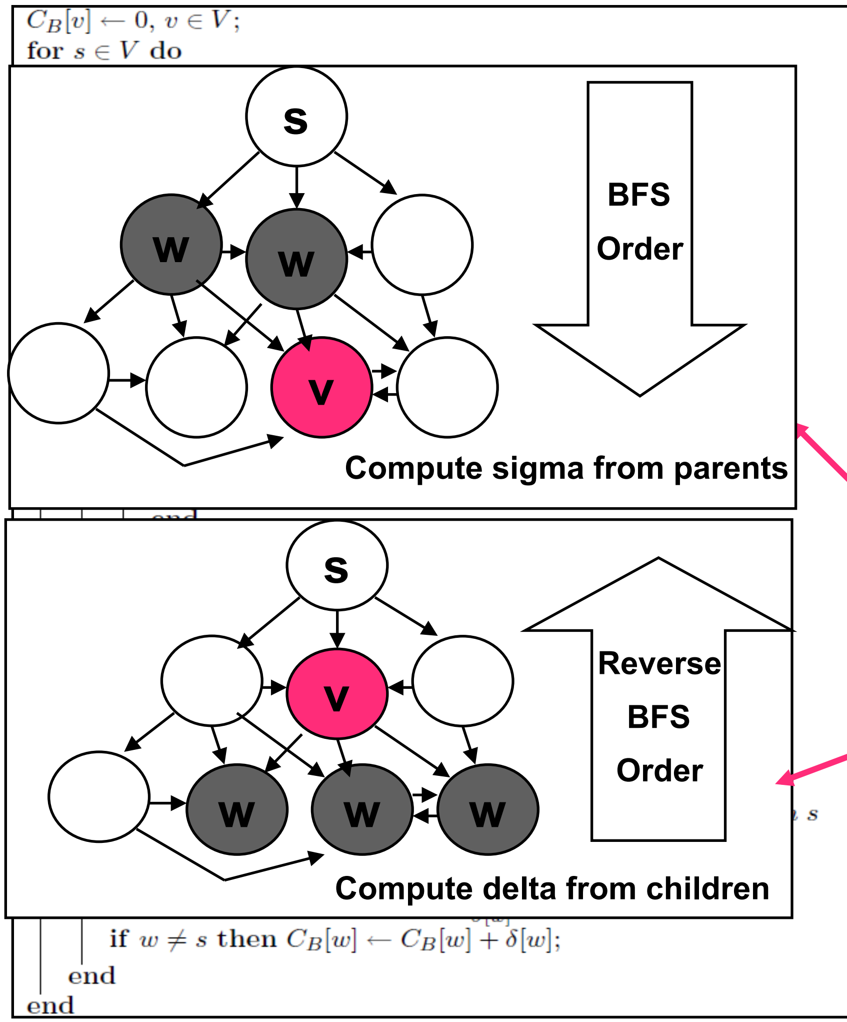
[Image source; Wikipedia]

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

ORACLE

Betweenness Centrality in a graph DSL

[Brandes 2001]




```
Procedure comp_BC(G: Graph, BC: Node_Property<Float>(G))
{
  G.BC = 0; // Initialize
  Foreach (s: G.Nodes) {
    // temporary values per Node
    Node_Property<Float>(G) sigma;
    Node_Property<Float>(G) delta;

    G.sigma = 0; // Initialize
    G.delta = 0;
    s.sigma = 1;

    // BFS order iteration from s
    InBFS(v: G.Nodes From s) {
      v.sigma = // Summing over BFS parents
        Sum (w:v.UpNbrs) {w.sigma};
    }
    // Reverse-BFS order iteration to s
    InRBFS(v:G.Nodes To s)(v!=s) {
      v.delta = // Summing over BFS children
        Sum (w:v.DownNbrs) {
          v.sigma / w.sigma * (1+ w.delta) };

     v.BC += v.delta @ s; // accumulate BC
    }
  }
}
```

Parallel Assignment

Parallel Iteration

Parallel BFS

Reduction

ORACLE