The Case for the Holistic Language Runtime System

Martin Maas*   Krste Asanovic*   Tim Harris†   John Kubiatowicz*

*University of California, Berkeley   † Oracle Labs Cambridge

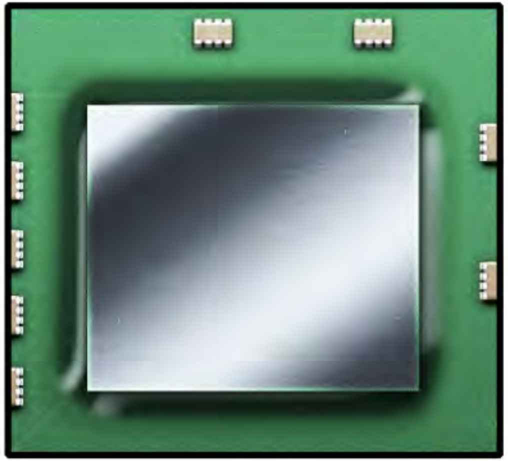
Cloud Data Centers in 2020

Trend to Rack-scale Machines

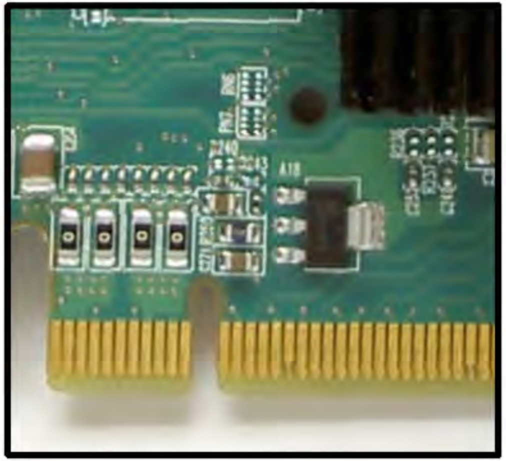
AMD SeaMicro

HP Moonshot

Rack-Scale Machines in 2020



Custom SoCs



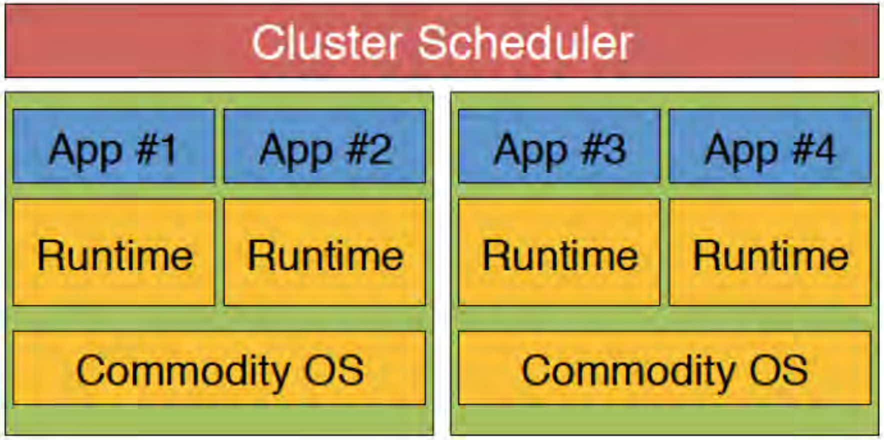
Flat low-latency Interconnects

Bulk NVM Storage

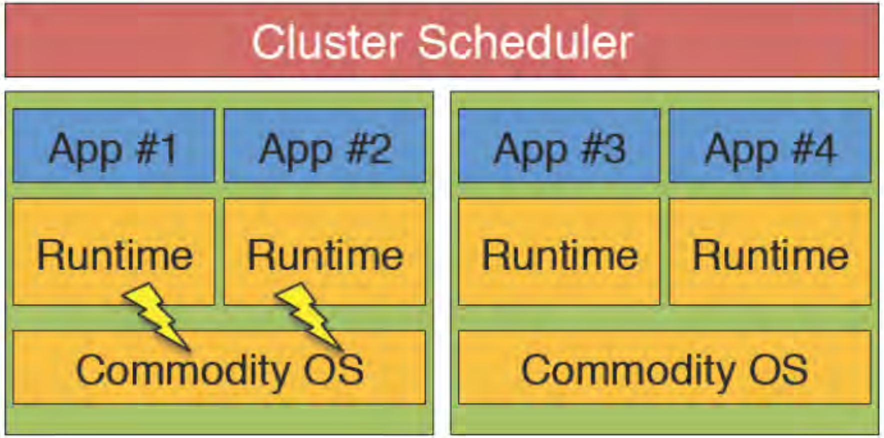
How will they be programmed?

Managed languages are everywhere!

Today's Software Stack



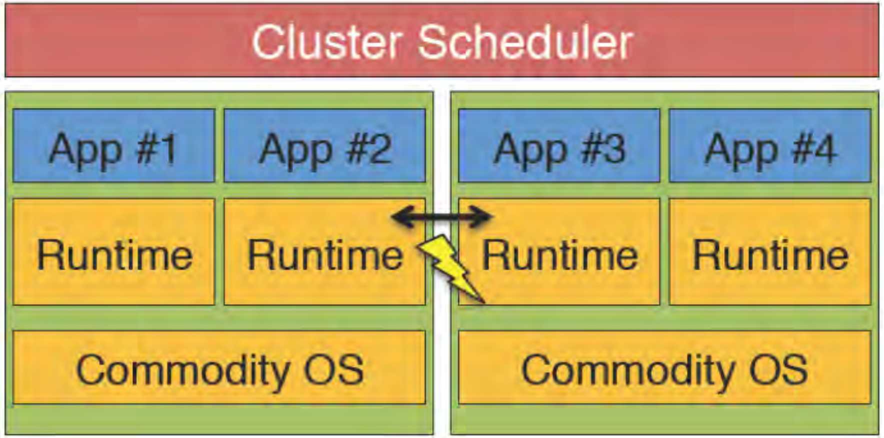
Problems with Today's Stack



Intra-node Interference

Problems with Today's Stack

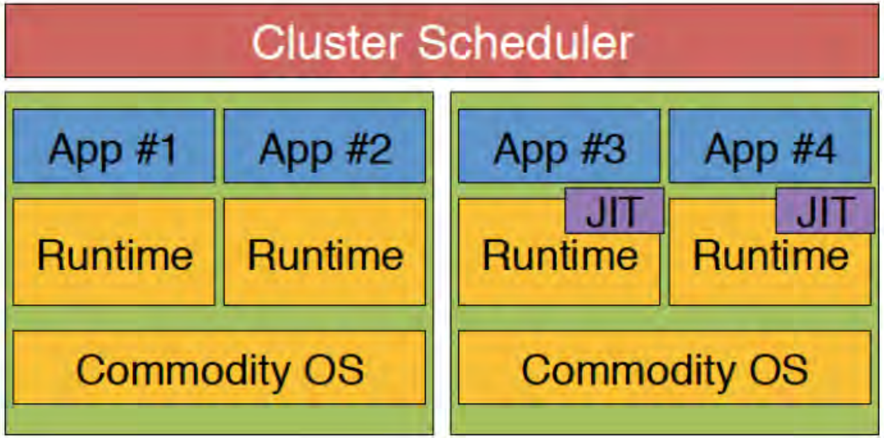
Intra-node Interference



Inter-node Interference

Problems with Today's Stack

Intra-node Interference



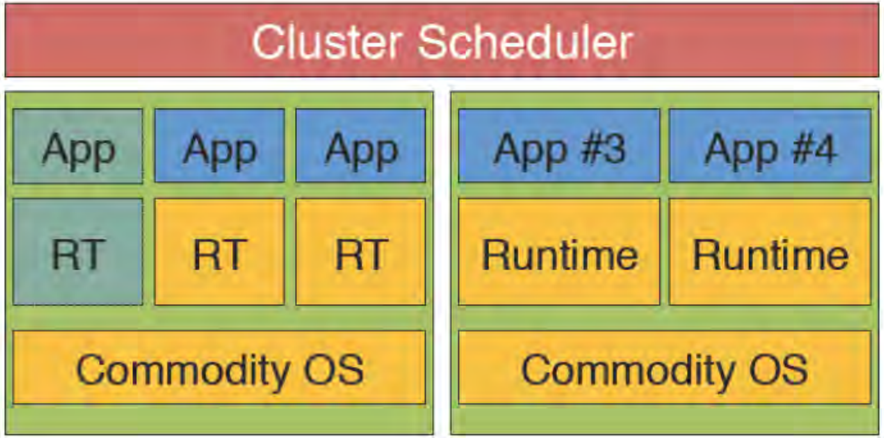
Redundancy

Inter-node Interference

Problems with Today's Stack

Elasticity

Intra-node Interference



Redundancy

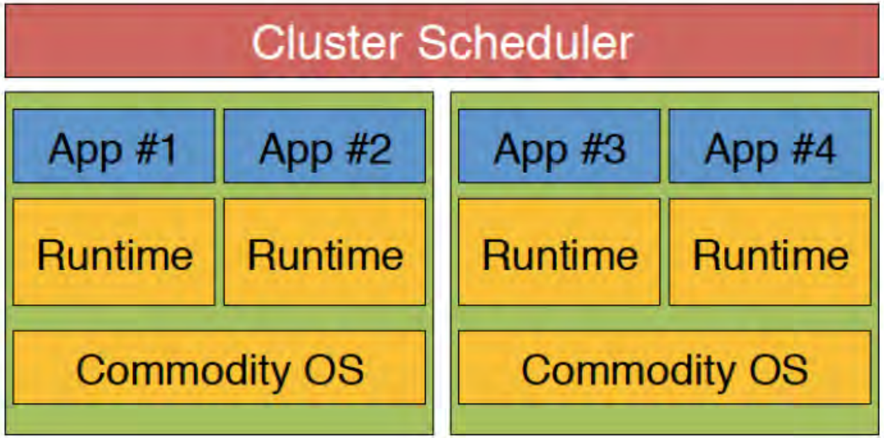
Inter-node Interference

We need a new approach

Restructuring the language runtime system and OS for rack-scale machines

Elasticity

Intra-node Interference

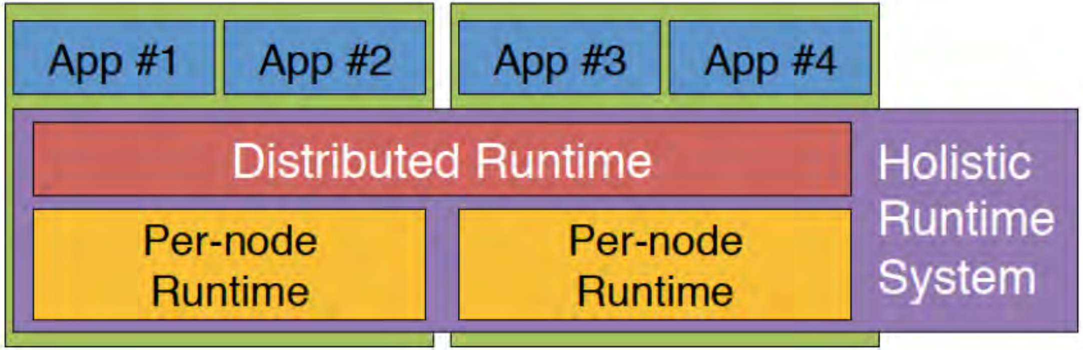


Redundancy

Inter-node Interference

We need a new approach

Restructuring the language runtime system and OS for rack-scale machines

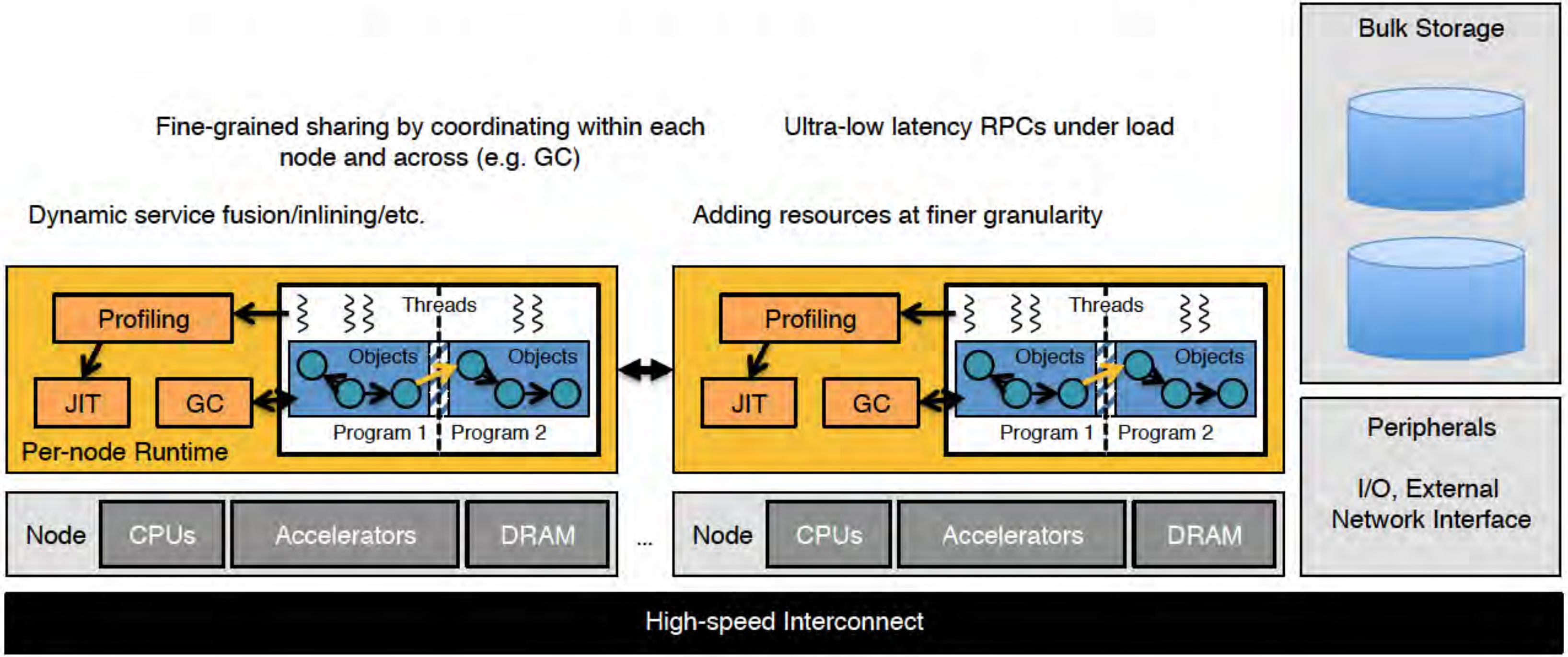


Talk Outline

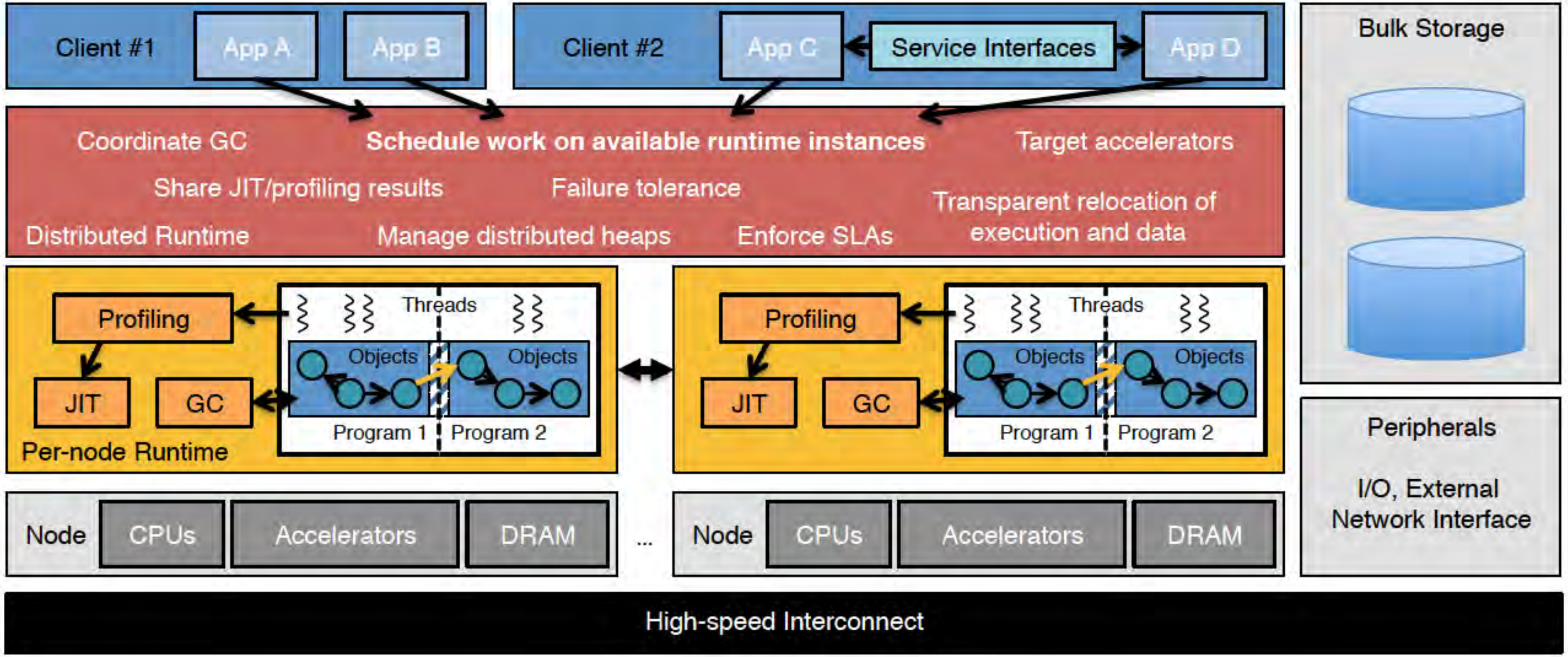
1. **Holistic Runtime Systems**
   Details, Advantages, Programming Model
2. **Cloud Data Center Trends**
   Holistic Runtimes tackle the challenges of 2020
3. **Challenges & Future Work**
   Research Directions & Opportunities

PART I
Holistic Runtime Systems

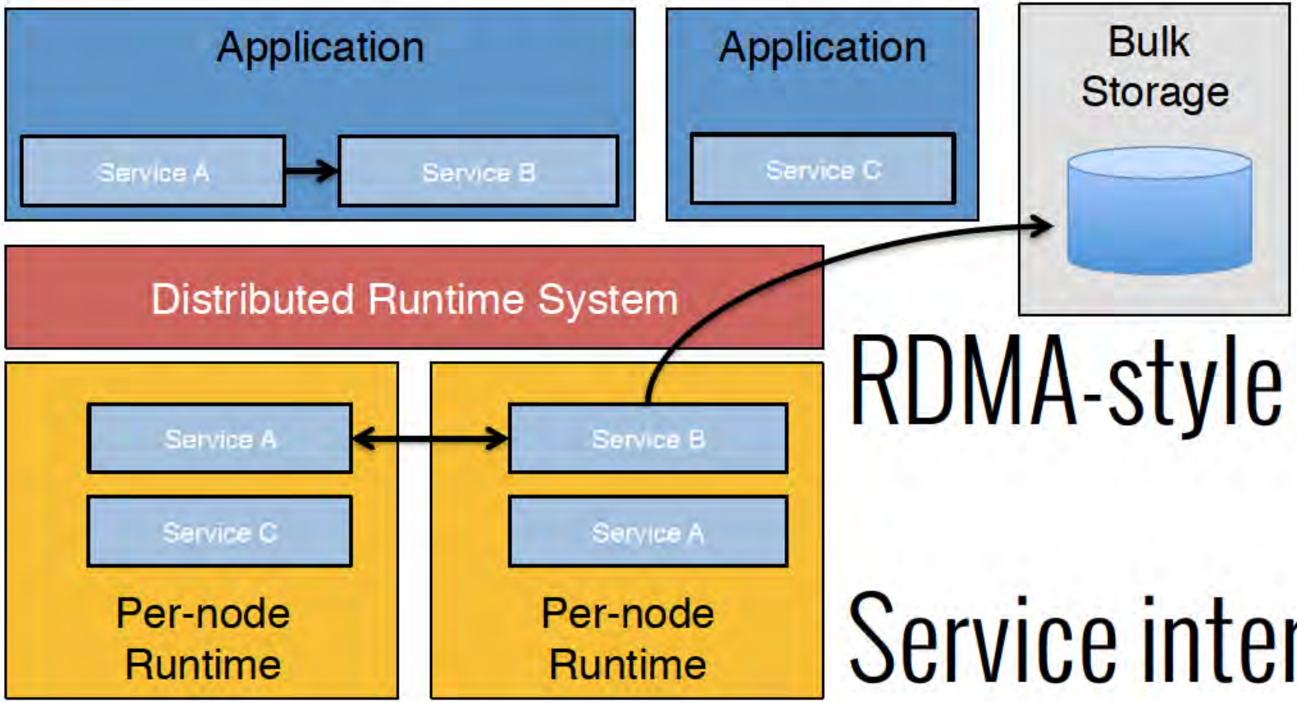
Holistic Runtime Systems



Holistic Runtime Systems



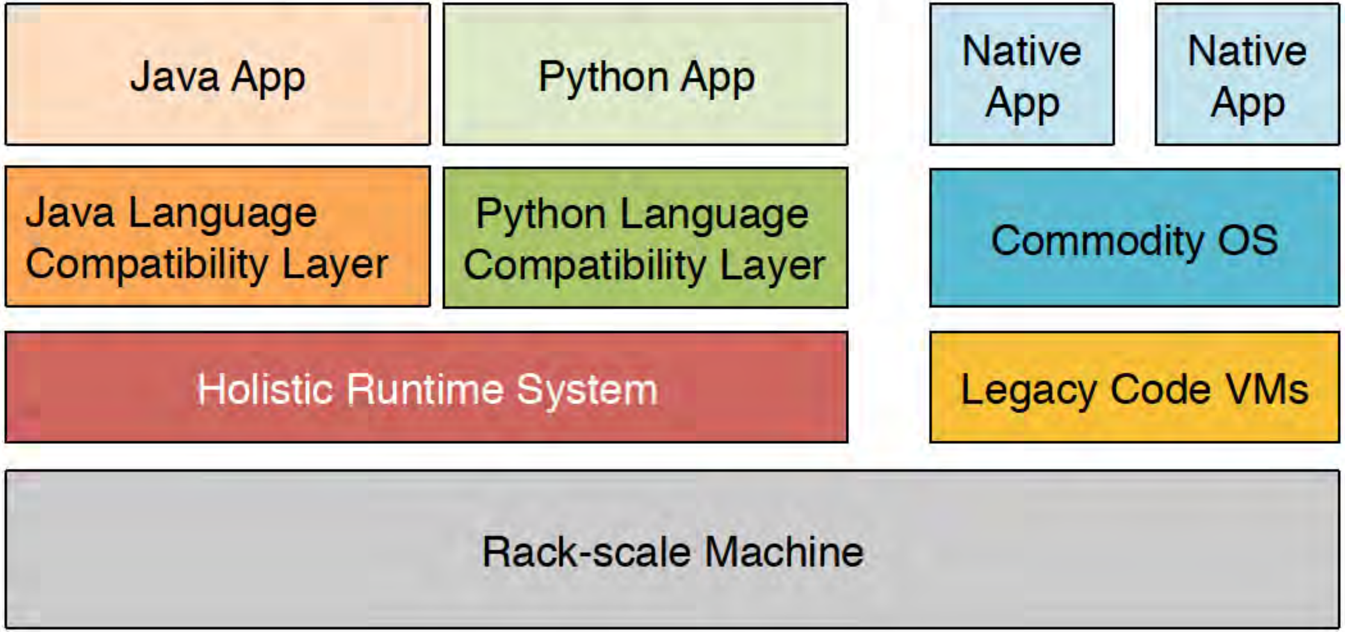
Programming Model



RDMA-style transfers

Service interfaces

Legacy Application Support



Java App

Python App

Native App

Native App

Java Language Compatibility Layer

Python Language Compatibility Layer

Commodity OS

Holistic Runtime System

Legacy Code VMs

Rack-scale Machine

Applicability

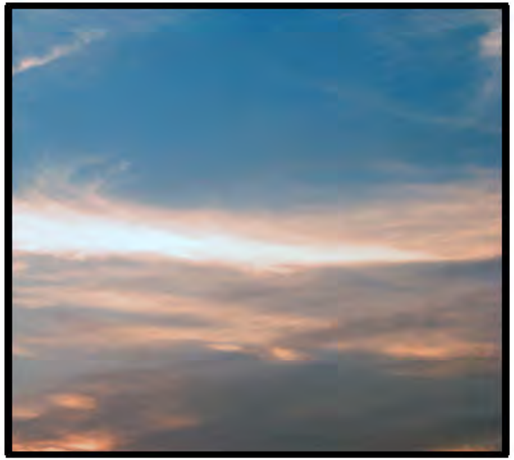
- Would be useful today to run managed workloads on Infiniband clusters
- But really benefit from tightly coupled nodes and hardware support
- Excellent fit for future data centers

PART II
Cloud Data Center Trends

Cloud Forecast for 2020

Hardware

Workloads

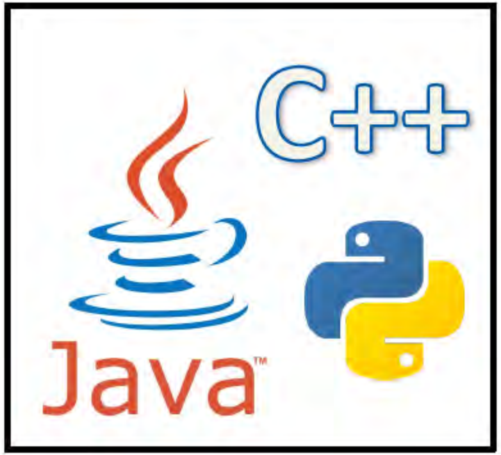


Languages

Cloud Data Centers in 2014

Hardware

Workloads



Languages

Cloud Data Centers in 2014

- Custom machines from commodity parts
- Redundant/unused components (I/O interfaces, peripherals, etc.)
- Inefficiency in energy and hardware cost

Hardware

Workloads

Languages

Cloud Data Centers in 2014

- Mostly developed in-house (e.g. Hotmail)
- Some interactive, mostly batch jobs
- Interleaved with external workloads

Hardware Workloads Languages

Cloud Data Centers in 2014

- Workloads written by mixture of systems programmers and domain experts
- Mix of native and managed languages (external workloads often managed)

Hardware Workloads Languages

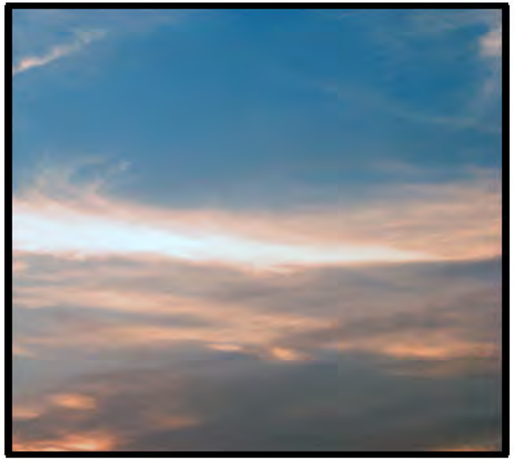
Critical Workloads in 2014

- Tune workloads to underlying cluster
- Provision lightly loaded nodes for jobs with low-latency requirements
- Write latency-critical applications in native languages (usually C++)

Cloud Data Centers in 2020

Hardware

Workloads

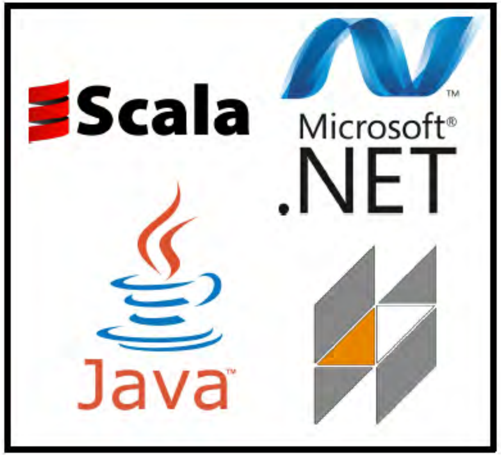


Languages

Cloud Data Centers in 2020

Hardware

Workloads



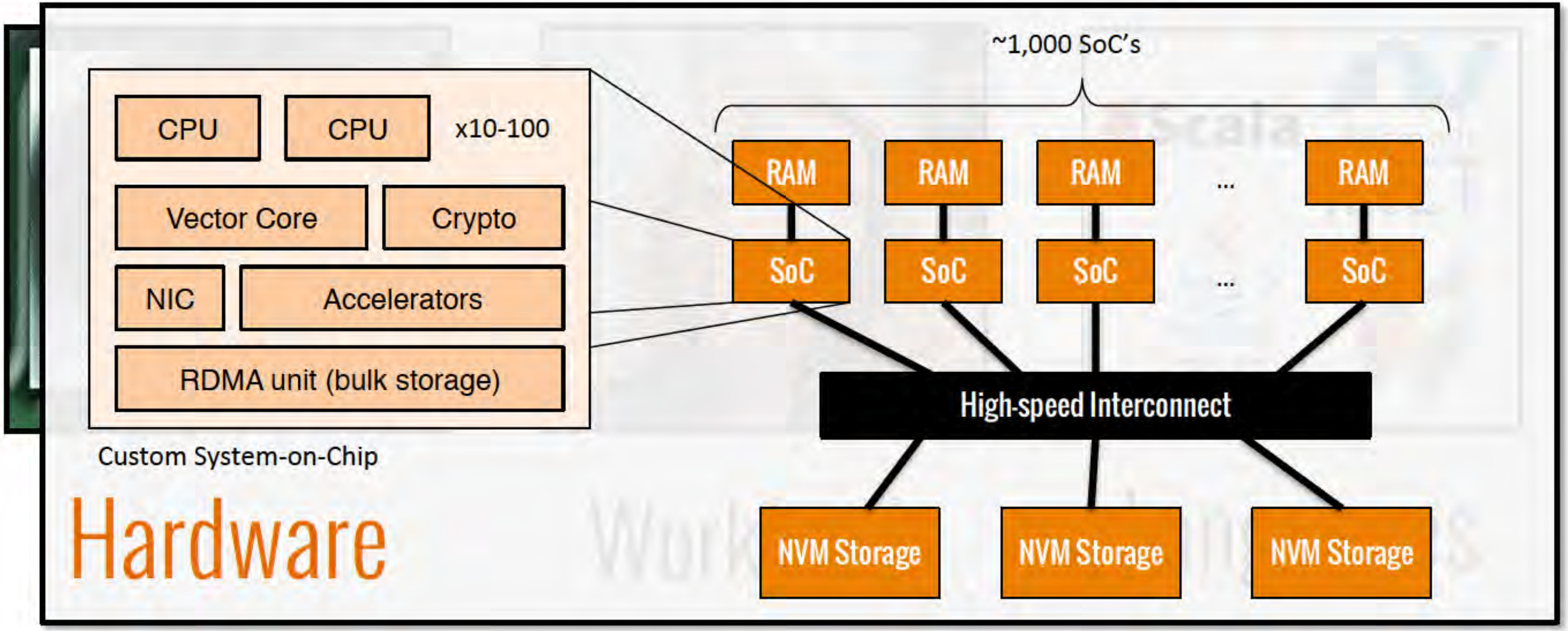
Languages

Cloud Data Centers in 2020

- Volume of cloud market growing → economically feasible to design custom SoCs
- Need to reduce energy and hardware cost
- Software benefits from hardware support

Hardware Workloads Languages

Cloud Data Centers in 2020

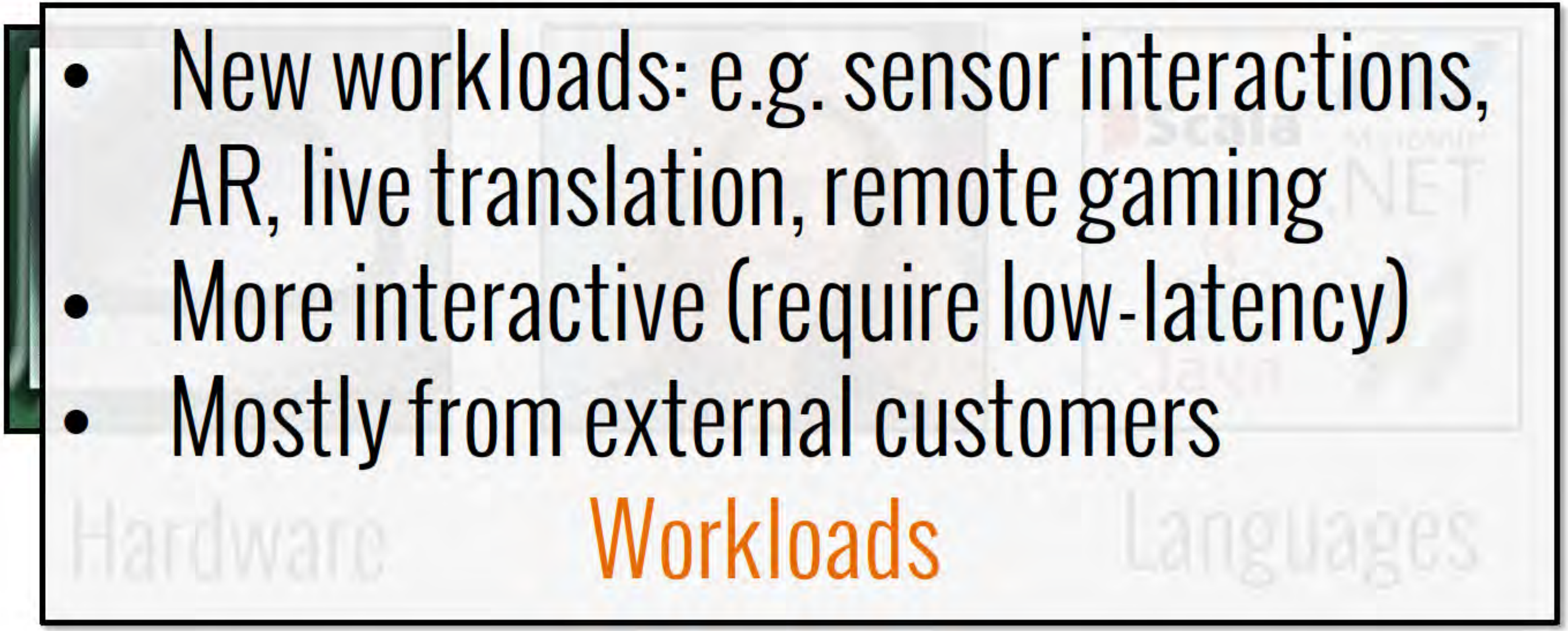


Custom System-on-Chip

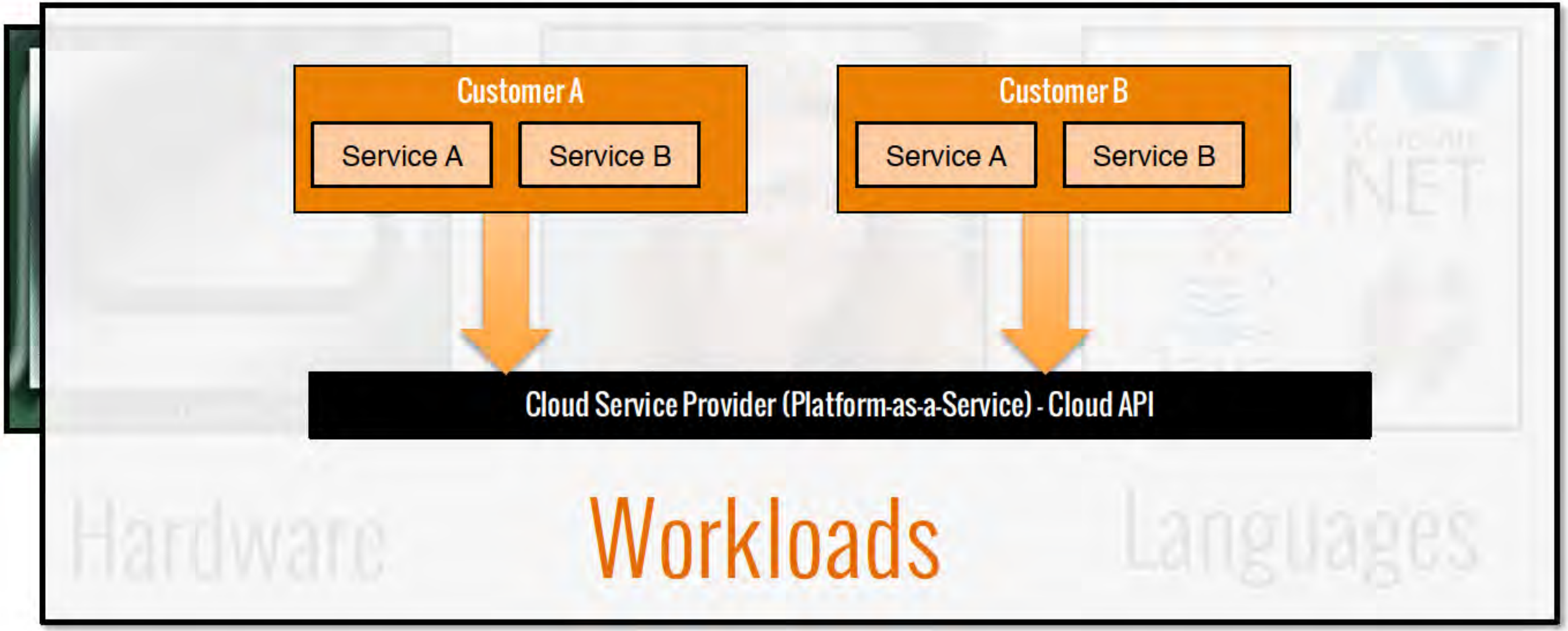
Hardware

Cloud Data Centers in 2020

- New workloads: e.g. sensor interactions, AR, live translation, remote gaming
- More interactive (require low-latency)
- Mostly from external customers



Cloud Data Centers in 2020



Cloud Data Centers in 2020

- Mostly written by external application developers (cloud will be main platform)
- Will almost exclusively use high-level languages and frameworks

Hardware Workloads Languages

Implications

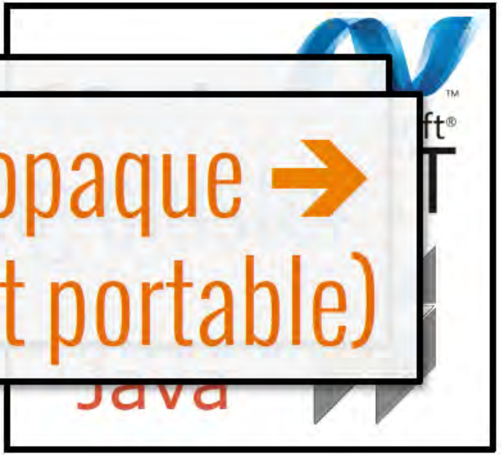
Tune applications the underlying cluster

Rack-scale machines

Mostly interactive

Productivity languages

Implications



The Cloud is becoming more opaque → fine-tuning infeasible (and not portable)

Rack-scale machines

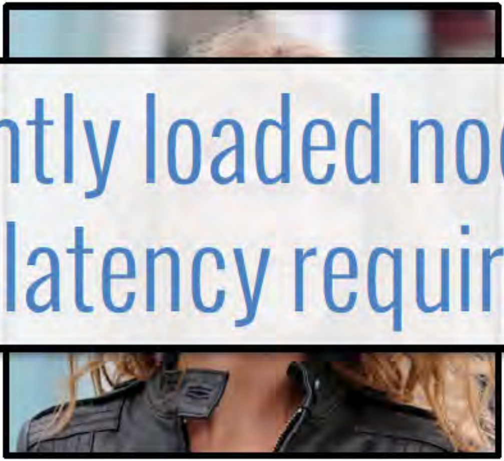
Mostly interactive

Productivity languages

Implications

Provision lightly loaded nodes for jobs with low-latency requirements

Rack-scale machines



Mostly interactive

Productivity languages

Implications

Radical over-provisioning will cease to be cost-effective ➔ sharing

Rack-scale machines

Mostly interactive

Productivity languages

Implications

Write latency-critical applications in native languages (usually C++)

Rack-scale machines

Mostly interactive

Productivity languages

Implications

Cloud will be exclusively programmed with high-level languages

Rack-scale machines

Mostly interactive

Productivity languages

Cloud Workloads written in managed languages, latency-sensitive and not tuned to the underlying platform

Holistic Runtime Systems exploit rack-scale machines to run them efficiently

PART III
Challenges & Future Work

Garbage Collection

- Garbage Collection of **tera- or peta-byte sized heaps** unsolved problem
- Bulk+local storage (e.g. *RAMCloud*)
- Cross-node references ➔ Distributed Garbage Collection

Fault Isolation

- Faulting application or SoC must not bring down rack-scale machine
- Isolation/lifecycle support in Java: JSR-121, Multi-tasking VM
- Potential for HW support (*Mondriaan*)

Performance Guarantees

- Probabilistic performance and tail latency guarantees for service calls
- High-level goals (e.g. *Tessellation OS*)
- Need predictable GC performance (HW support is work in progress)

Conclusion

Conclusion

- Cloud data centers are changing:
  - Rack-scale machines, interactive/external workloads, managed languages
  - Current software stack is a bad fit
- Are Holistic Runtimes the solution?

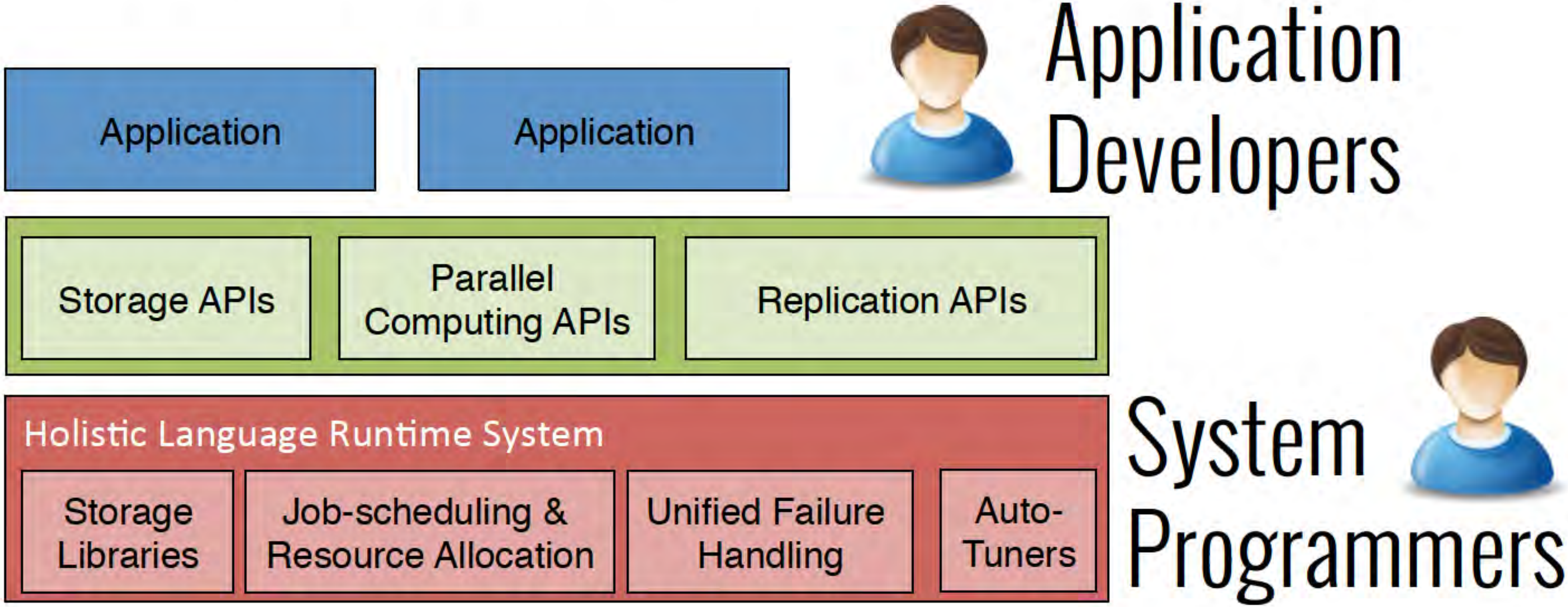
Thank you! Any Questions?

Martin Maas, Krste Asanovic, Tim Harris, John Kubiatowicz

maas@eecs.berkeley.edu, krste@eecs.berkeley.edu,
timothy.l.harris@oracle.com, kubitron@eecs.berkeley.edu

Backup Slides

Frameworks & Extensibility



Why Managed Languages?

- Much better productivity and safety
- Abstract away hardware details and can transparently tune to platform
- Semantics allow fine-grained sharing
- Good for service-oriented architecture

Programmability Crisis

- Productivity programmers...
- ...programming for an increasingly complex but opaque platform...
- ...with strict latency requirements under high sharing of machines

Problems with current stack

- Current software stack is a bad fit:
  - Interference: Intra- and inter-node
  - Redundancy: JIT, class library, etc.
  - Composability: RPC latencies
  - Elasticity: Start-up/boot times