Pradeep Sundarrajan, Abhishek Gupta, Mathew Ryden,
Abhishek Chandra, Jon Weissman

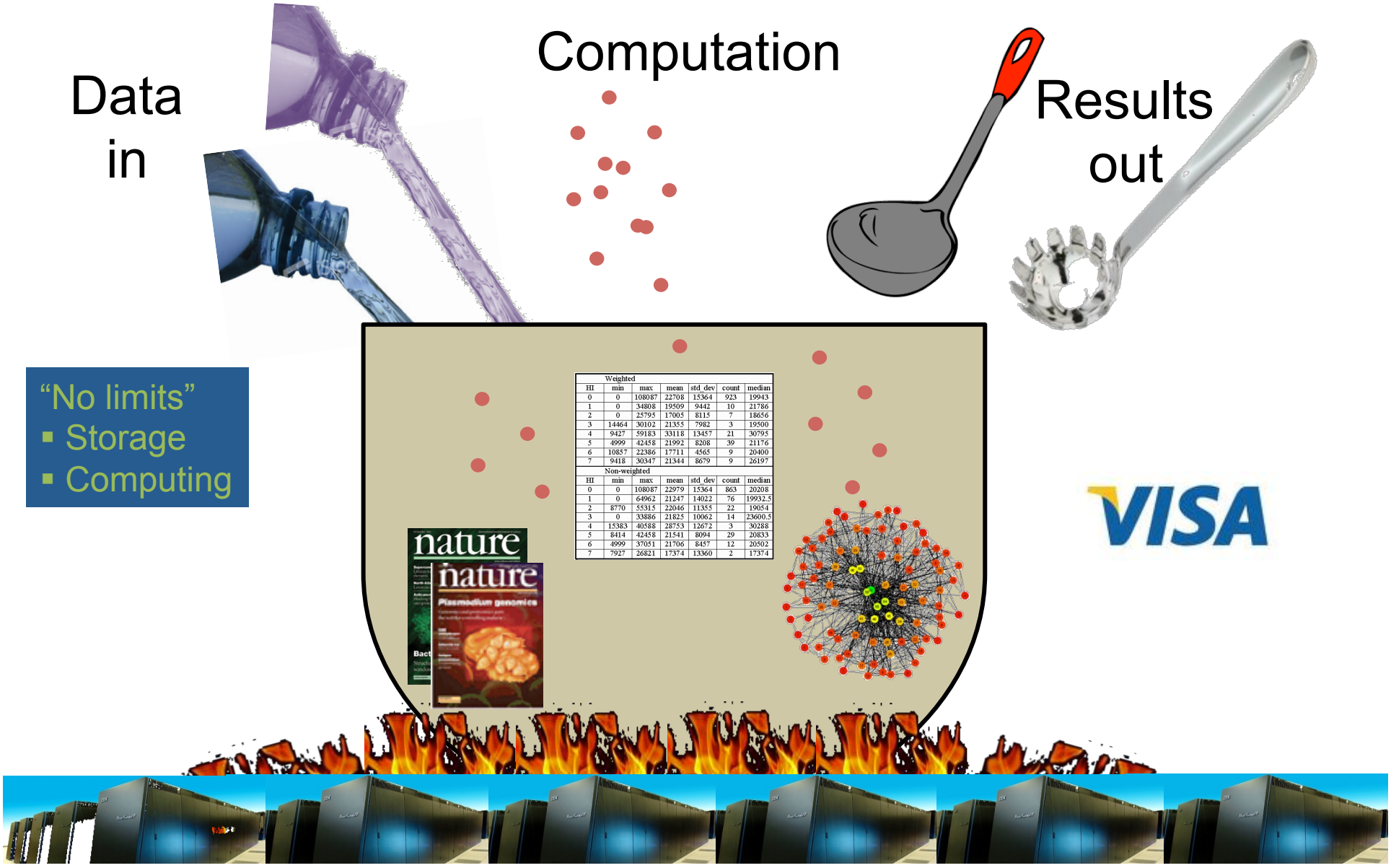
Department of CS&E

University of Minnesota

Outline

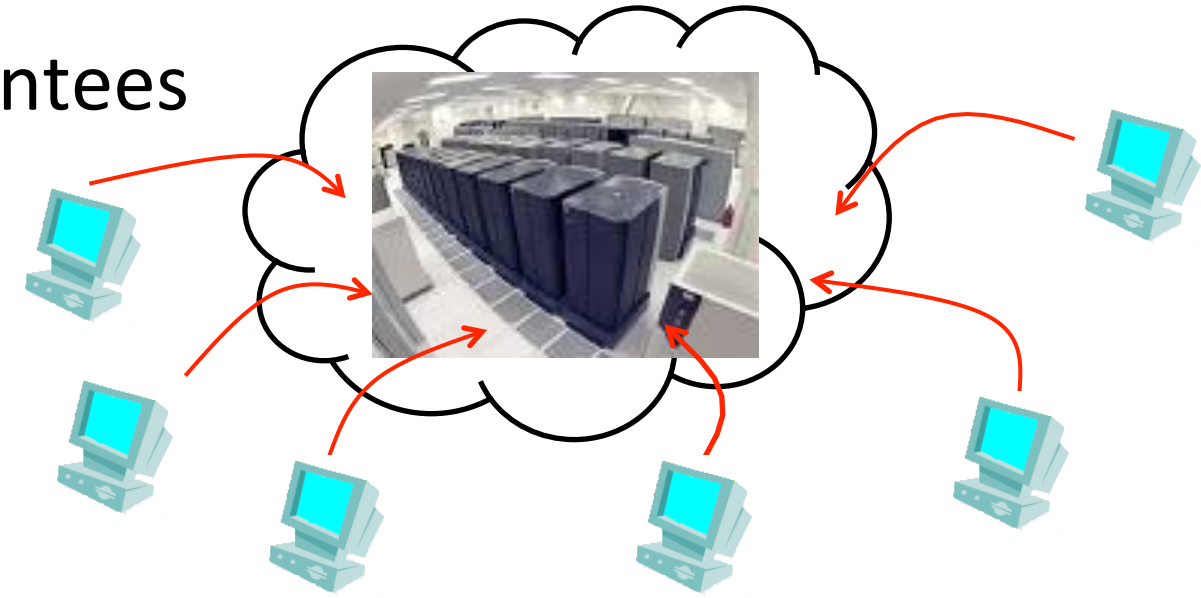
- Conventional cloud
- Limitations and opportunities
- Nebula project

The “Standard” Cloud



Current Cloud Model

- Largely centralized
- Pay-as-you-go
- Strong guarantees
- 3rd party



Appealing Features

- Scale/consolidation
  - elasticity, lower TCO
- Strong locality
  - data and computing => great for analytics
- Novel sharing platform
  - data/state and applications => gaming, Web 2.0

Fraying at the Edges

- Privacy
  - don't want everything going to the cloud but some things
- Social/community networks
  - limited sharing
- Locality
  - largely centralized cloud => bottlenecks
    - to users …
    - to/from data sources … (think: Big Data)

Big Data Trend

- Big data is distributed
  - earth science: weather data, seismic data
  - life science: GenBank, NCI BLAST, PubMed
  - health science: GoogleEarth + CDC pandemic data
  - web 2.0: user multimedia blogs
  - “everyone is a sensor”

Privacy/Locality Trend

- Privacy
  - restrict/filter data (think: patient records)
- Locality
  - mobile users: latency sensitive application access
  - criticality: "deliver go-signal to my insulin pump"

Need New Features

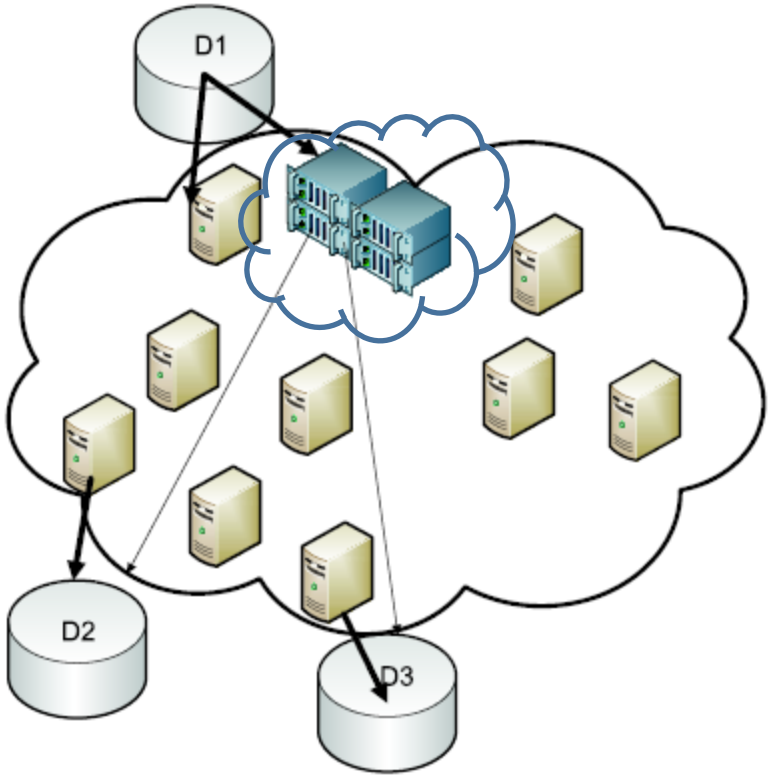
- Process data in-situ or close by
  - save time and money
  - privacy-aware
- Organize platform based different notions of "closeness"
  - network distance
  - trusted nodes
  - social groups
  - communities of interest

Idea

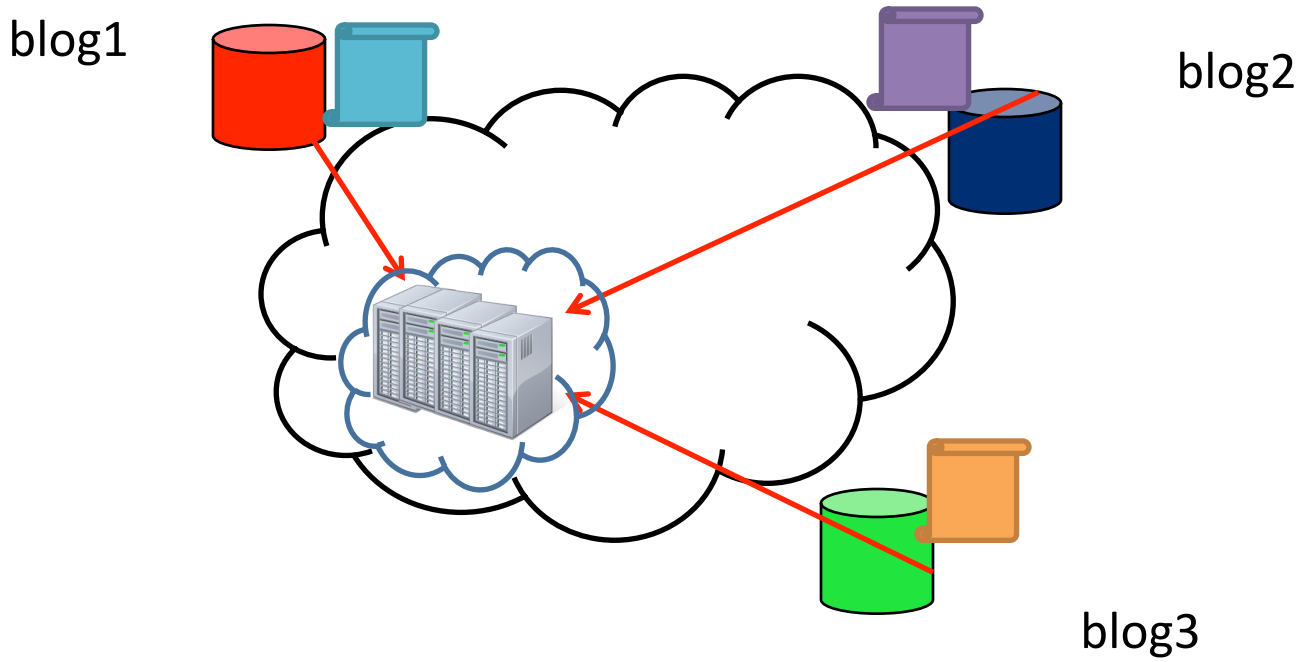
- Make the cloud more “distributed”
  - “move” it closer to data
  - “move” it closer to end-users
  - “move” it closer to other clouds

Example: Dispersed-Data-Intensive Services

- Data is geographically distributed
  - Costly, inefficient to move to central location

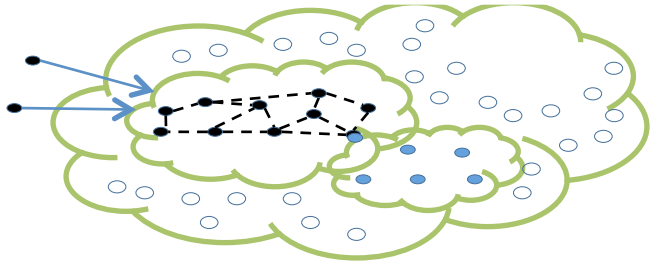


Example: Blog Analysis



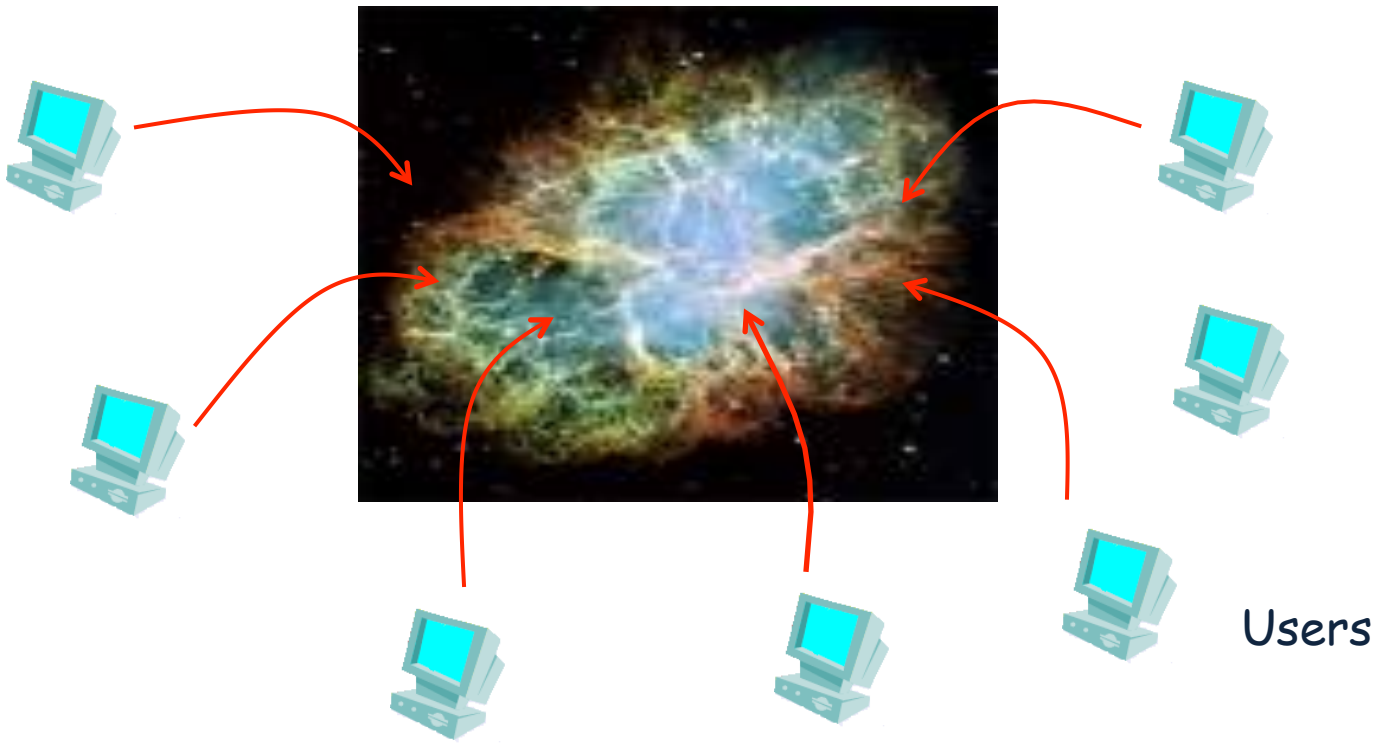
Nebula: A New Cloud Model

- Stretch the cloud
  - exploit the rich collection of edge computers
  - volunteers (P2P, @home), commercial (CDNs)

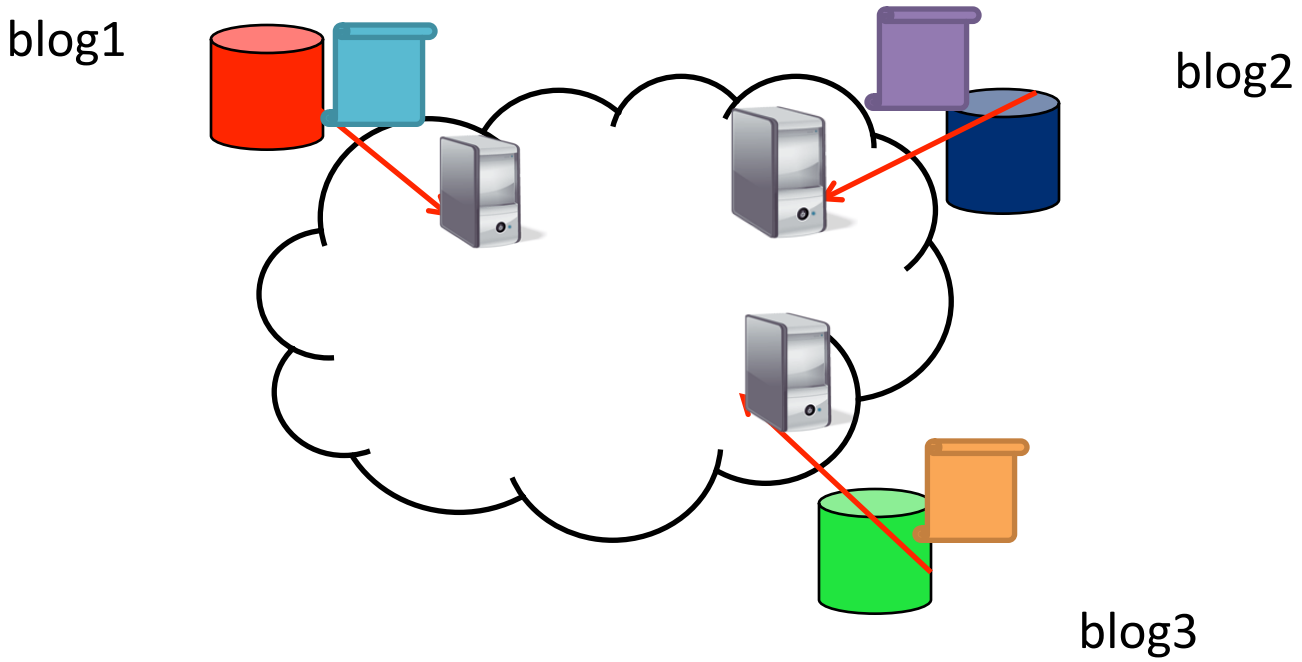


Nebula

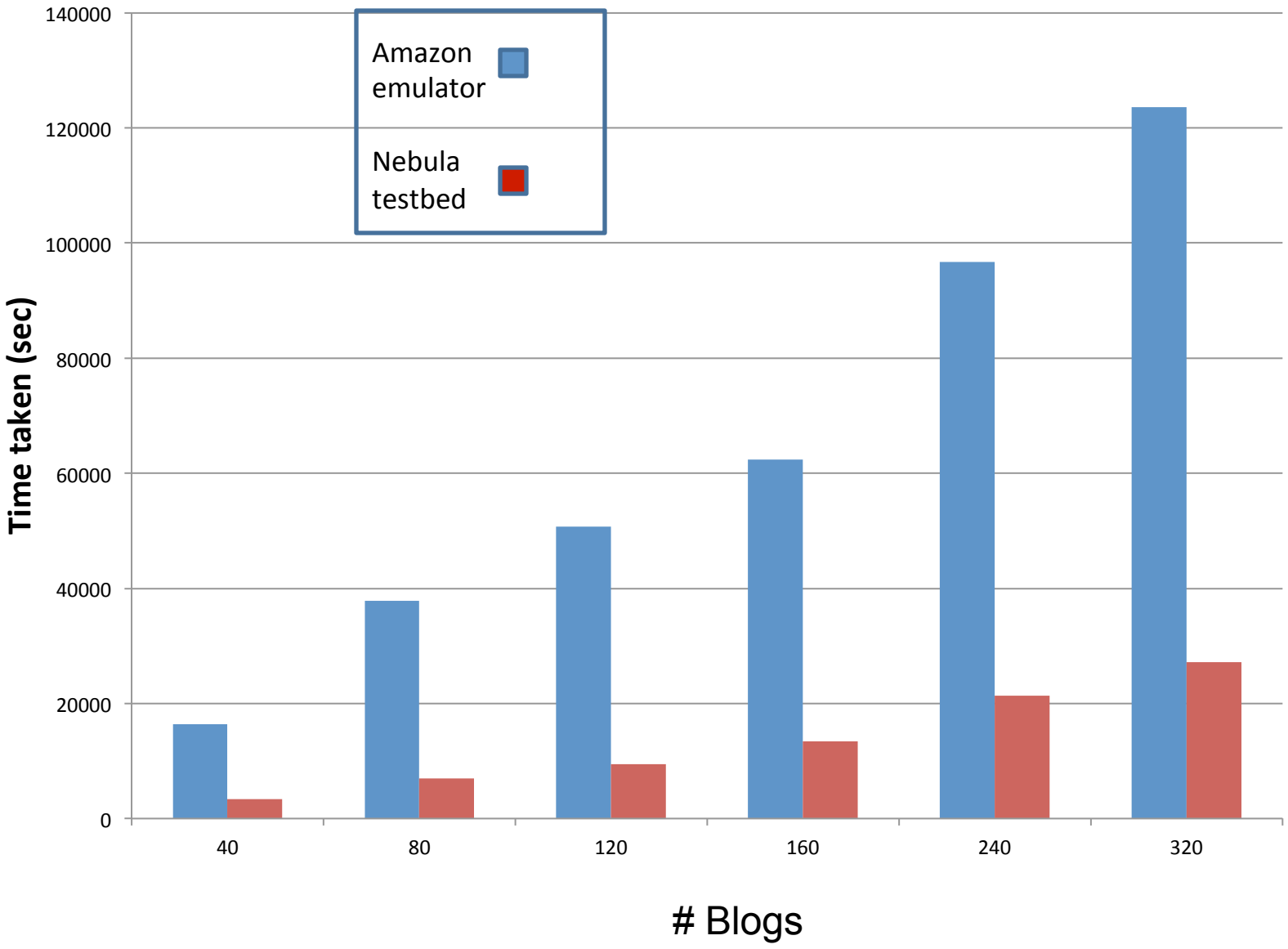
- Decentralized, less-managed cloud
  - dispersed storage/compute resources
  - low user cost



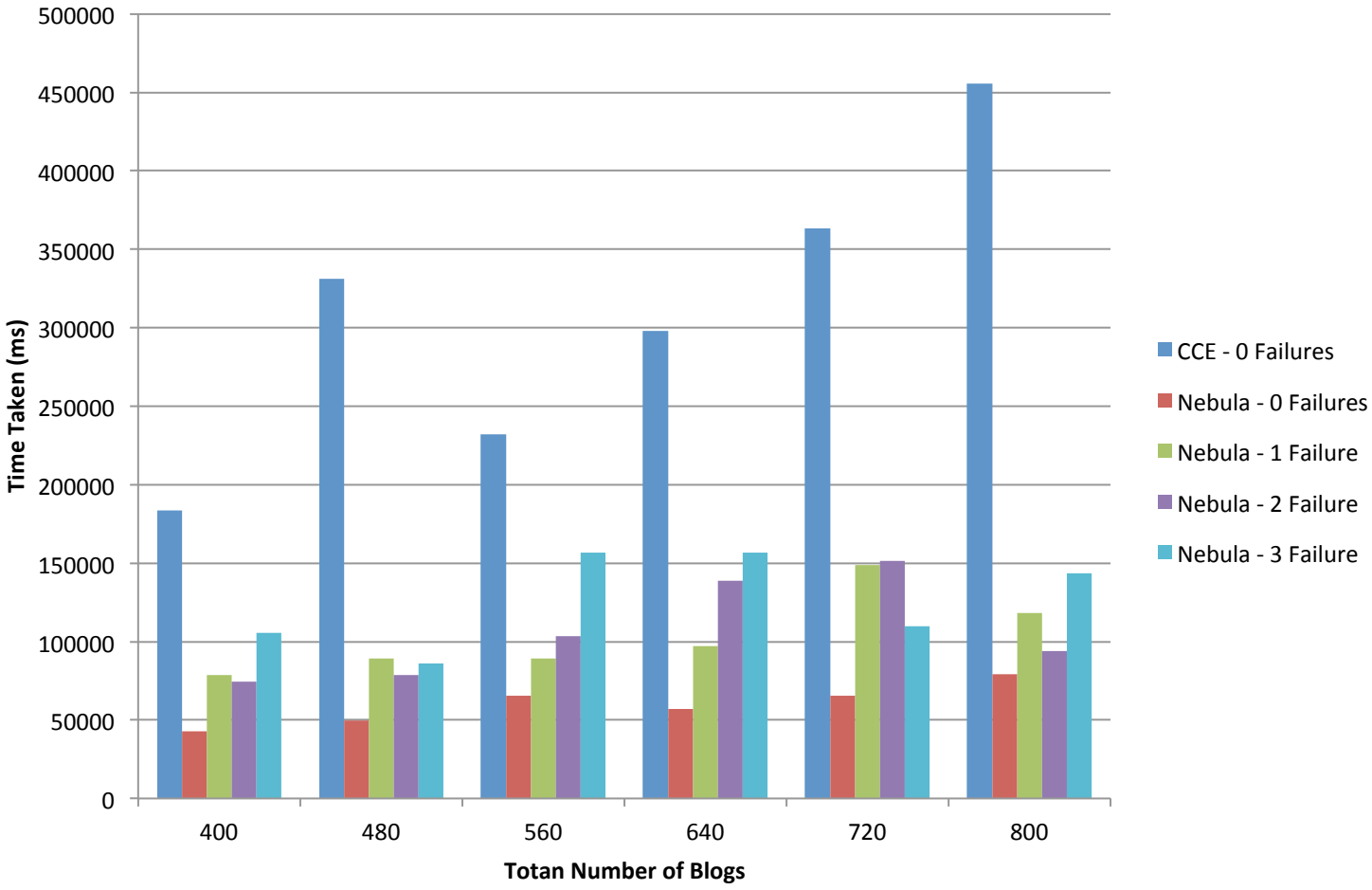
Example: Blog Analysis



Blog Results

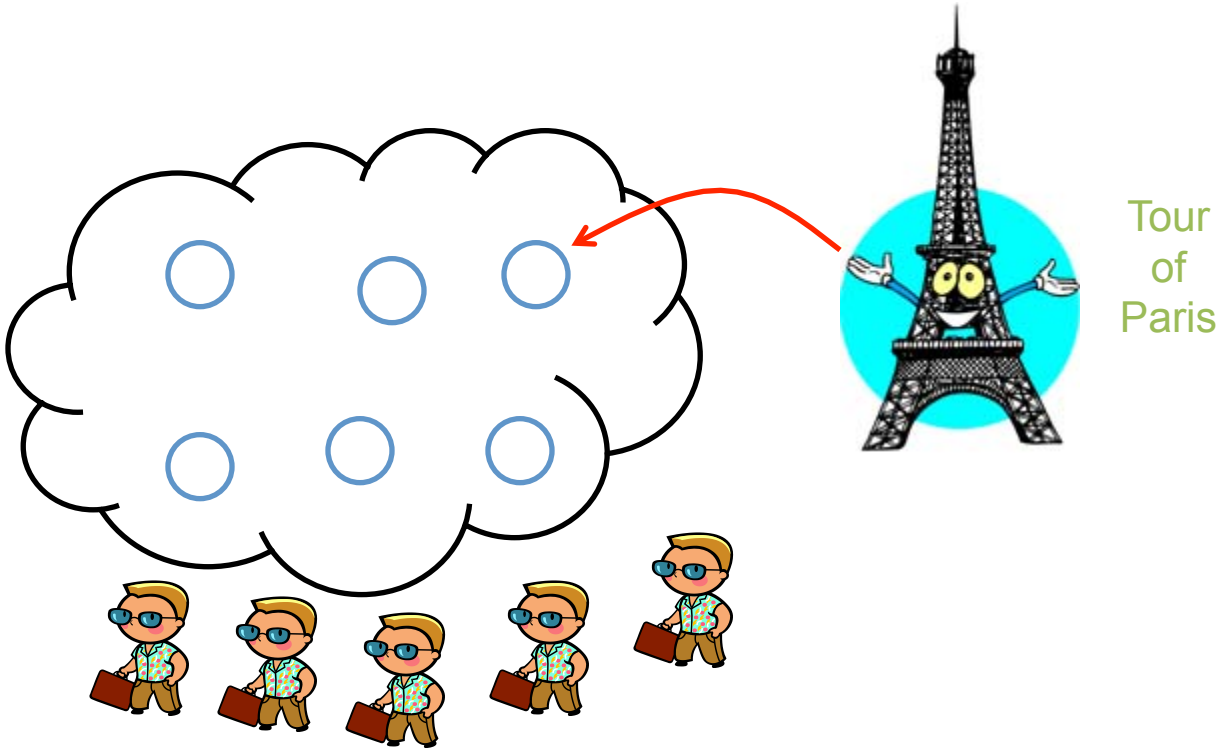


Failure Resistant



Another Example: Latency-Sensitive

- Mobile service



How is Nebula different from @home?

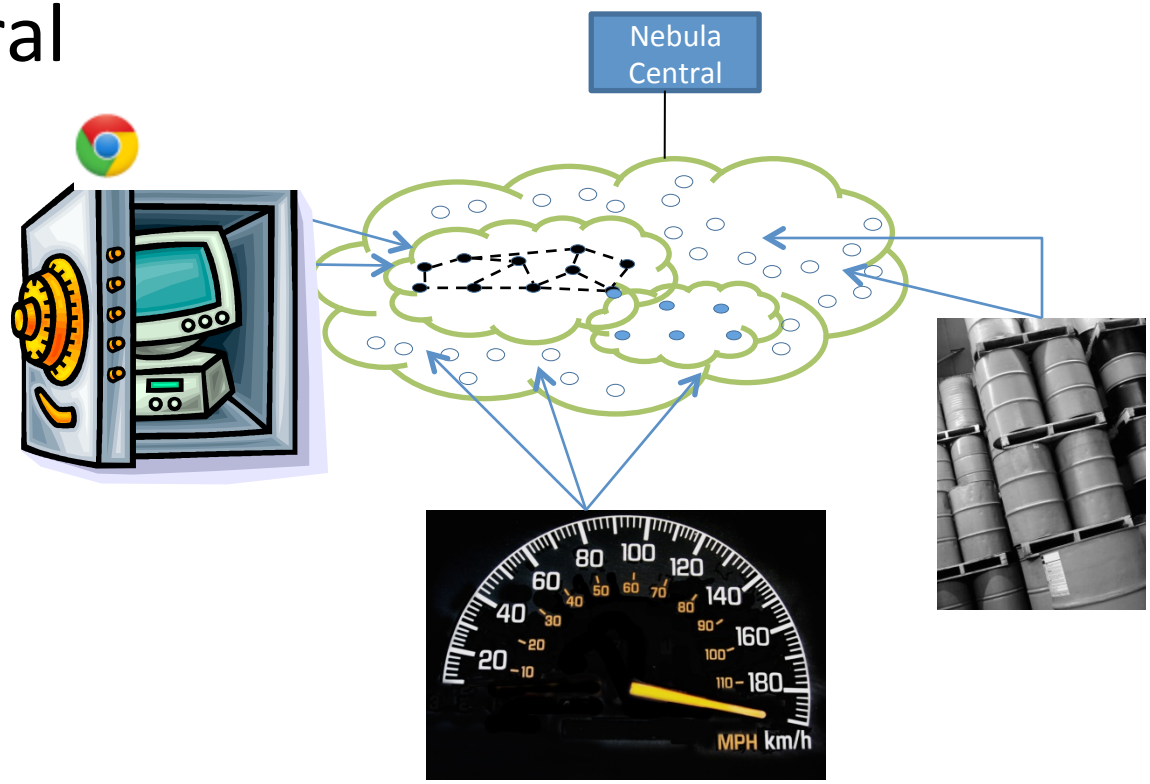
Requirement	Nebula	@home
Collective performance	High	None
Locality/Context-awareness	High	Low
Statefulness	High/medium	Low

Common Service Characteristics

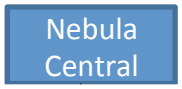
- Elastic resource consumption
  - scale up/down based on demand
- Geographical data/user distribution
  - execution dependent on location of data/user
- Weak performance/robustness requirements
  - some failures may be tolerable

Inside Nebula

- Nebula central
- Chrome
- Dashboard
- Datastore



Nebula Central



- Manager
- Volunteers check-in
- Tracks global state of other services
- Distributes code and nebula software
- Run at UMn
- Central point of trust

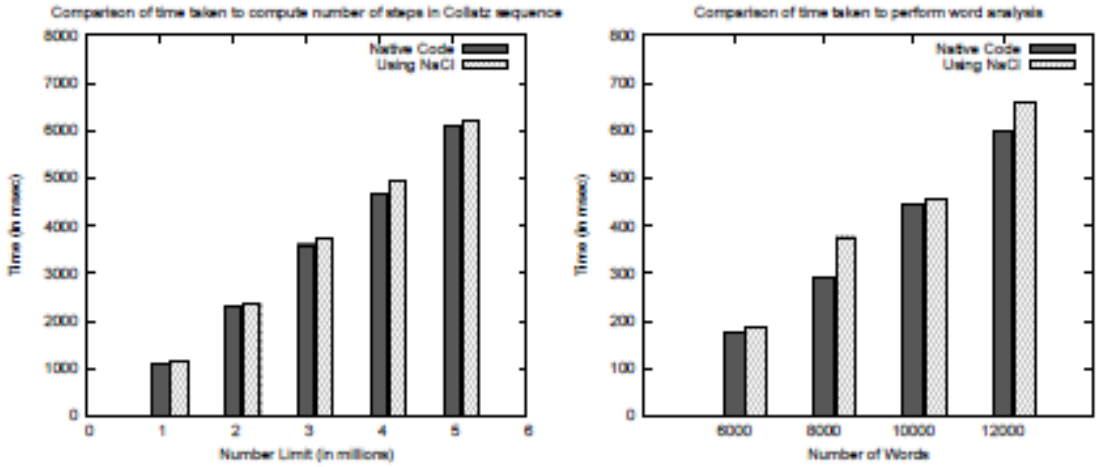
DataStore

- Data service that runs on subset of nodes
- Provides basic store/retrieval
- Policy-based management for a specific DS
  - capacity, latency, fault tolerance, durability

Operation	Parameters	Description
client_create_group	datastore_id, source, destination, scoring_function, number	Create data node group
client_put	source, datastore_id, file_id, number	Put a file in a data group
client_get	datastore_id, file_id	Fetch a file from a data group

SecureNode

- Nebula nodes run a Chrome Browser
  - secure sandbox (NaCL) native client inside
  - all native code executes inside it



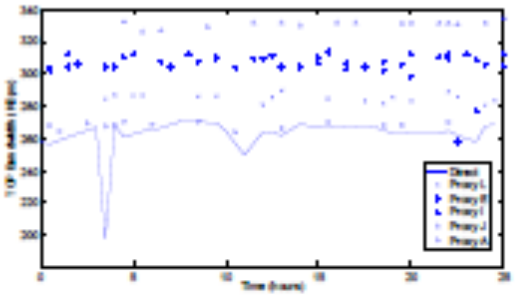
(a) Collatz (b) Text analysis

Figure 11: Performance overhead of NativeClient

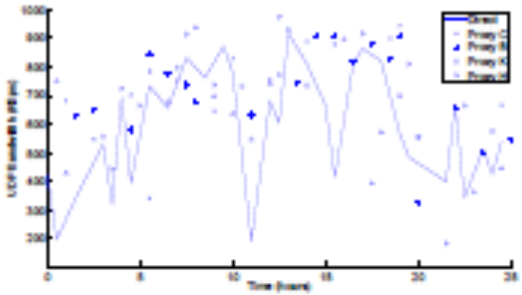
Network Dashboard

- Software tool `netstat.cs.umn.edu`
- Runs on all nebula nodes
- Provides point-to-point latency, jitter, bandwidth
- Used by DataStore service, NodeGroup service (future)

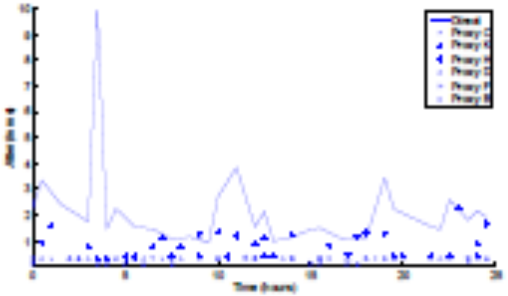
Dashboard Output



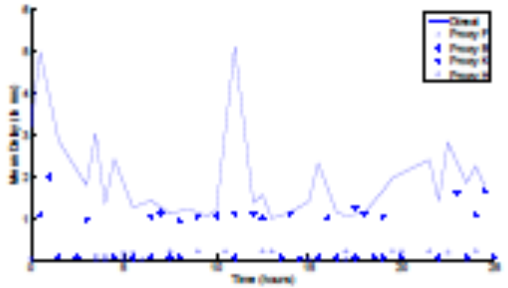
(a) TCP bandwidth (KBps) (Higher is better)



(b) UDP bandwidth (KBps) (Higher is better)



(c) UDP delay (ms) (Lower is better)



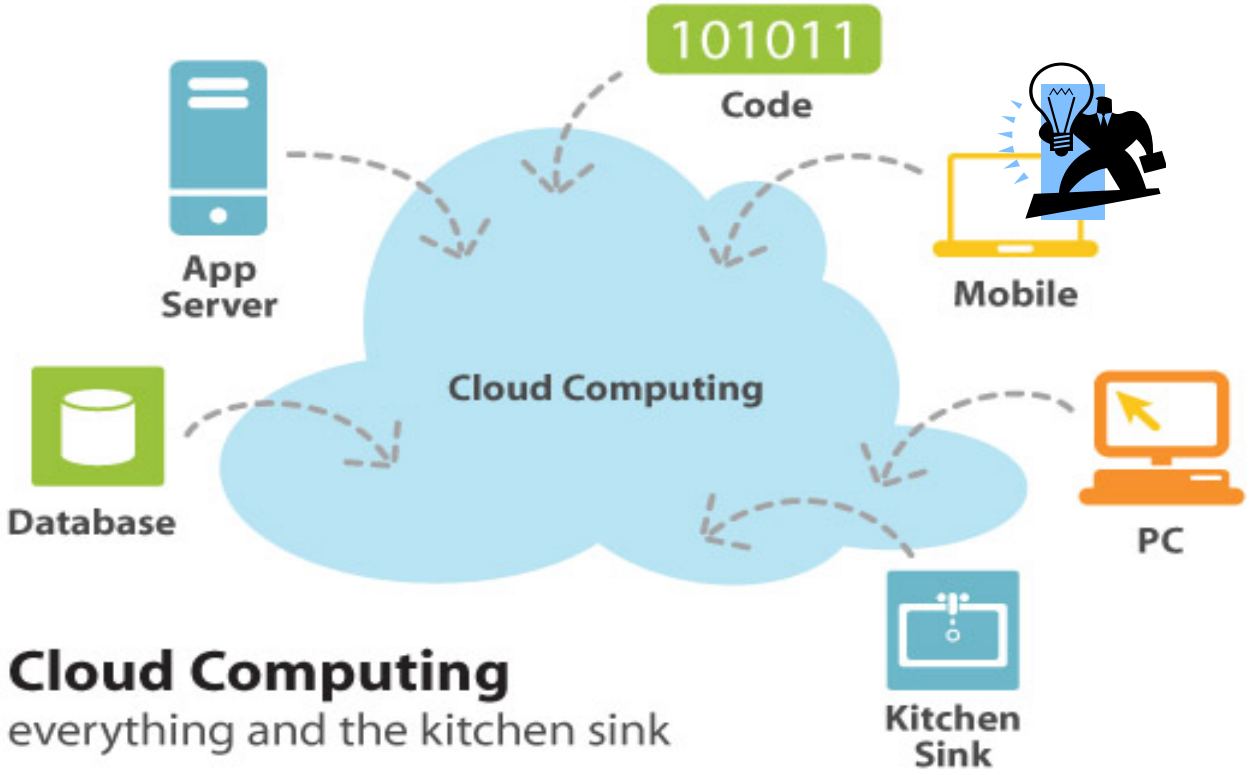
(d) UDP jitter (ms) (Lower is better)

Summary

- Nebula: new cloud architecture
  - Preserves cloud behavior: APIs, elasticity, transparency
  - Stronger notion of external locality
  - Weaker notion of internal locality

- Future work
  - End-to-end system operational
  - Connect to the commercial cloud

    "use the edge opportunistically"



Thank you! Questions?