Final Project Presentations

• Presentation Logistics
  – Where: CSE 403
  – When 1pm-5pm on Monday, March 16th
  – Use the Doodle to pick a slot
  – Check course website for detailed schedule
  – Each team gets 10 min
    • 8 min to talk and 2 min for questions (strictly enforced)

• Please try to attend all the talks

• Grading
  – One grade for the whole team
  – But I recommend that each team member speaks at least briefly
Recommended Presentation Format

- **Introduction and problem statement (2min)**
  - What problem are you trying to solve?
  - Mention your related work here
  - Introduce a running example and use it in the rest of the talk

- **Approach (3min)**
  - How are you trying to solve the problem?
  - Why did you pick this solution and not another?

- **Preliminary evaluation (2min)**
  - What did you find out?
  - What are your results?

- **Conclusion and future work (1min)**
Final Project Report

Write your final report like a mini conference paper
• Abstract: 1 short paragraph
• Introduction: 1 page [The intro is a summary of the paper]
• Section 2: Background, problem, or motivation: 0.5 page
  – Describe the application domain, the data, the problem
• Section 3: Approach: 2 pages
• Section 4: Evaluation: 2 pages
• Section 5: Related work: 0.25 page
• Section 6: Conclusion: 1 paragraph
Length: Aim for 6 pages but ok to spill over
Course Evaluation

https://uw.iasystem.org/survey/141298

• Did you find the homeworks useful?
  – HW1: I already got some comments: change the dataset, offer an alternative to installing PostgreSQL, spend less time on data munging.
  – HW2 and HW3? Any suggestions?
• Did you like the selection of lecture topics?
  – Good mix of background/fundamentals and more recent topics?
• Did the project help you to learn about data management and the research process?
• Overall workload? Was it ok?
• Did you like having a guest speaker?
• Any other comments are welcome!

Thanks!
References

Cloud Computing

• A definition
  – “Style of computing in which dynamically scalable and often virtualized resources are provided as a service over the Internet”

• Basic idea
  – Developer focuses on application logic
  – Infrastructure and data hosted by someone else in their “cloud”
  – Hence all operations tasks handled by cloud service provider

• Some history
  – "computation may someday be organized as a public utility” (John McCarthy – 1960)
  – 1996 Hotmail “Software as a Service”
  – 1999 Salesforce.com offers enterprise-class “Software as a Service”
  – 2006 Amazon Web Services with EC2
  – And now it’s commonly used
Levels of Service

- **Infrastructure as a Service (IaaS)**
  - Virtual machines, storage, and networking
  - Example Amazon EC2

- **Platform as a Service (PaaS)**
  - Execution runtime, database, web server, development tools, …
  - Example Microsoft Azure, Google App Engine

- **Software as a Service (SaaS)**
  - Entire applications
  - Example Google Docs
How About Data Management as a Service?

- **Running a DBMS is challenging**
  - Need to hire a skilled database administrator (DBA)
  - Need to provision machines (hardware, software, configuration)
  - Problems:
    - If business picks up, may need to scale quickly
    - Workload varies over time

- **Solution: Use a DBMS service**
  - All machines are hosted in service provider’s data centers
  - Data resides in those data centers
  - Pay-per-use policy
  - Elastic scalability
  - No administration!
Multitenancy Problem

- Given a DBMS as a cloud service, how to support multiple tenants?

1. Each tenant runs in its own virtual machine(s)
   - For example Amazon RDS

2. Tenants share the same DBMS instances
   - For example SQL Azure

3. Tenants data is stored in a single table
   - For example Force.com
Tenant Placement

• Many tenants need less than the capacity of one machine

• How to consolidate many tenants on a few servers?
  – Also called “tenant packing”

• Question 1: Which tenants can be placed together?
  – Want to avoid interference
  – One challenge is that tenant workloads vary over time

• Question 2: How many tenants can we place together?
  – Trade-off between over-provisioning and over-booking
Tenant Migration

• When conditions change and SLOs are violated

• Need to move tenants
  – Which tenant to move?
  – How to perform the migration with minimum disruption?
Paper Discussion

• Delphi: Self-managing controller for a multitenant DBMS

• Pythia: Learn behavior through observation
  – Tenant behavior
  – Node behavior
  – Uses database-level attributes
  – Assigns a class to each tenant and determines which tenant classes can be colocated
  – Assigns classes to packings: good, good with underutilized resources, or bad
Tenant Model

• **DBMS-agnostic database-level performance measures**
  – Write percent (insert, delete, updates)
  – Avg operation complexity: avg nb of pages accessed by tx
  – Percent cache hits
  – Buffer pool size: nb pages allocated to tenant
  – Database size
  – Throughput (transactions per second)

• **Tenant labels**
  – D: Disk IOPS, T: Throughout, and O: Operation complexity
  – Each resource type range is split into buckets
  – Tenant labels: DS-TS-OS
Node Model

• One feature per node: packing vector
  – One cell per tenant class
  – Value in cell is the number of tenants of that class

• Model learns mapping
  – From feature vector
  – To quality of packing: under, good, over
Crisis Detection and Mitigation

- Periodically collect a snapshot of system state
- For each snapshot, classify tenants
  - Tenant class is aggregate class over time-window \( W \)
    - Example: \( \{0.8c_j, 0.2c_k\} \)
- If packing is bad, use hill-climbing to find a good packing
  - Consider all potential migrations of one tenant
  - Perform the move that yields the largest improvement
    - Naïve cost function minimizes the number of nodes labeled as “over”
      - Not good because algorithm tends to overload one node completely
    - Better cost function assigns a confidence to each node of being over
      - Consider only nodes with high confidence of being over
      - Minimize the weighted sum of tenants being on an overloaded node
  - Continue until cannot improve any more