> Klarstellung Wimpynodes

IPs:

- 141.76.47.218 Controller (← hier bitte keine DB-Prozesse starten und keine Daten ablegen)
- 141.76.47.220-235

Directories:

- /home/gruppe<X>
  - NFS-mount on controller -> same on every machine
  - install your DB and code here, so it is available on all machines
- /media/localdisk/gruppe<X>
  - local 40GB SSD (roughly 20 GB free at the moment)
  - create your data directories here, so that nodes write/read on local disk

Please install all components of our experiment on the same Node for now…

- the DB server
- your application
- but also the database generator and the workload definitions (setup.db / wldriver-r)

… and run the workload driver from the same machine

Postgres with normalized schema, application in Python (Flask/Gunicorn), running on one Wimpy Node

Load 1000, 1 Node:

- users 4.4s
- followers: 11.6s
- tweets: 164.8s

Run 1000, 1 Node:

- ~1800 T / 30s
- 1,6s / Q

Run 1000, 1 Node, only 10 threads:

- ~1800 T / 30s
- 0,15s / Q

Run 1000, 1 Node, only 1 thread:

- ~900 T / 30s
- 0,03s / Q

You can vary the number of threads in wldriver-r/config.cfg

It should be possible to get results in the same order of magnitude with each DBMS used

Some tips if you're not satisfied with your results:

- Start small: run and measure a single query against you application, e.g. with
  - curl -X POST -H "content-type: application/json" -w %{time_total} -d '{"username":"user_1"}' http://127.0.0.1:<your-port/<your-endpoint>
  - This is exactly the same as what the workload-driver does and will print the request time
- If your solution does a lot of work in the application (e.g. looping over the the user's "friends") try to profile your application
  - Simply printing time differences is often enough, e.g. in Java use System.currentTimeMillis()
- Try to find out which parts of your query processing actually take up the time
  - Is it really the DB? Is it something in your application?
- If most of the work is done in the database:
  - Check if your DB has an EXPLAIN-like feature, i.e., some feature to display what parts of your query take how long inside the DB
  - Some usual problems: the DB has to scan all the data for each query → add indices
  - Or the DB has to access a lot of tweets/users that are not needed → maybe you can restructure your data?
- Check if you are using you DB in the right way. This is different for each NoSQL DB, but there are guides, tutorials and examples out there
- For example, this is a nice general article on NoSQL data modeling patterns http://highlyscalable.wordpress.com/2012/03/01/nosql-data-modeling-techniques/

Die Zuordnung der angebotenen Praktika (und aller anderen DB-Lehrveranstaltungen) zu den Modulen der Bachelor- und Masterstudiengängen findet man unter

- http://wwwdb.inf.tu-dresden.de/student-info/pruefungen/moduluebersicht/
- Die hier beworbenen Praktika können sowohl in den Modulen 510/20/30/40 als auch als "Forschungspraktikum" abgerechnet werden (siehe folgender Tabellenausschnitt)

		SWS			Bachelor			Master			Diplom (PO 2010)	Diplom (PO 2004)
Vorlesung	WS/SS	V	Ü	P	Informatik	Medien-Informatik	Wirtschafts-Informatik	Informatik	Medien-Informatik	Wirtschafts-Informatik	Informatik	Informatik
...												
Komplexpraktikum DB-SyA	WS und SS	0	0	4	B-510, B-520	B-530, B-540			E-4			X
Komplexpraktikum DB-Anw	WS und SS	0	0	4	B-510, B-520	B-530, B-540			E-4			X
Forschungspraktikum DB-SyA	WS und SS	0	0	4				PM-FPG			PM-FPG	
Forschungspraktikum DB-Anw	WS und SS	0	0	4				PM-FPA			PM-FPA	

Hintergrund

Die Forschungsvision des Sonderforschungsbereiches HAEC (Highly Adaptive Energy-Efficient Computing) ist es energie-effizientes high-performance Computing zu ermöglichen. Momentane Infrastrukturen erlauben es nicht den Energieverbrauch eines Serversystems zu überwachen und zu kontrollieren. Daher ist es das Ziel des Teilprojektes EAST (Energy-Aware STream Configuration and Management) diesen Verbrauch zu überwachen, zu analysieren und vorherzusagen.

Dazu werden effiziente Analysen von Sensorwerten und Lastprofilen benötigt, welche kontinuierlich in die Datenbank strömen. Unter diesen Analysetechniken befinden sich einfache Verfahren wie beispielsweise Trendanalysen, aber auch komplexere Algorithmen zur Prognose zukünftiger Werte.

Aufgabe

Aufbauend auf dem bereits im Projekt EAST@HEAC eingesetzten Datenbanksystem DexterDB sollen im Zuge dieses Praktikums Datenbankoperatoren für die oben genannten Analysetechniken entwickelt und in die DexterDB integriert werden. Der mit den zu entwickelnden Operatoren abgedeckte Workload erstreckt sich von einfachen adhoc-Anfragen bis hin zur automatischen Aktualisierung und Wartung von Prognosewerten für Streamingdaten. Dabei zeigen sich die folgenden Aufgabenbereiche:

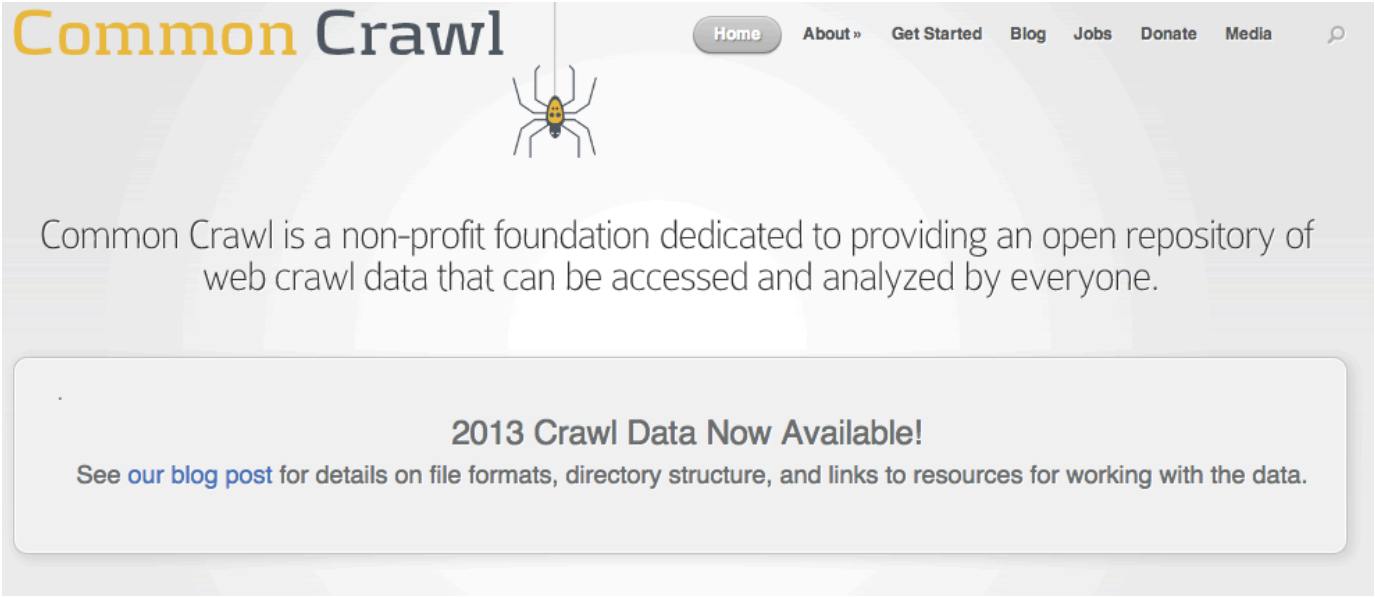
- Einarbeitung in das bestehende Datenbanksystem DexterDB (C/C++) und in die Grundlagen der Analysetechniken
- Entwurf der Datenbankoperatoren und deren Integration in das bestehende System
- Vergleichende Evaluation der entworfenen Operatoren mit anderen Ansätzen bzw. eine Performanzanalyse der Operatoren

Der letztendliche Umfang des Praktikums wird abhängig von der Teilnehmeranzahl durch die Betreuer festgelegt.

Kontakt

Dipl.-Ing. Thomas Kissinger - thomas.kissinger@tu-dresden.de

Dipl.-Inf. Claudio Hartmann - claudio.hartmann@tu-dresden.de



The coolest KP in town

- Work with TBs of data! (102 TB)
- Learn about and work with Hadoop, a mega-buzzword for the last several years!
- Learn about and work with structured data on the web! (~170 million data tables)!
- Disprove the results of famous researchers!

Contact

- Katrin Braunschweig (http://wwwdb.inf.tu-dresden.de/team/staff/dipl-medien-inf-katrin-braunschweig/)
- Julian Eberius (http://wwwdb.inf.tu-dresden.de/team/staff/dipl-medien-inf-julian-eberius/)