Development of Parallel DBMS on the Basis of PostgreSQL

Mikhail Zymbler, Constantin Pan
South Ural State University (SUSU)
Supercomputer Simulation Laboratory (SSL)
Chelyabinsk, Russia

Database Systems Research Group headed by Prof. Dr. Michael Gertz,
Institute of Computer Science
Ruprecht-Karl University Heidelberg, Germany

Wednesday, May 09, 2012
Outline

- Background
  - South Ural State University (SUSU)
  - SUSU's Supercomputer Simulation Laboratory
- PargreSQL, a parallel DBMS based upon PostgreSQL
  - Partitioned parallelism
  - PargreSQL architecture
  - Current results
South Ural State University
http://www.susu.ac.ru
South Ural State University

- **History**
  - 1943-1951: Chelyabinsk Mechanics and Engineering Institute
  - 1951-1990: Chelyabinsk Polytechnic Institute
  - 1990-1997: Chelyabinsk State Technical University
  - 1997-now: South Ural State University

- **Present**
  - 32 faculties
  - 2,700 professors and assistant professors
  - 55,000 students
  - 400 international students
  - 300 programs of higher professional education
  - 200 programs of further education

- **Achievements**
  - In top 10 Russian Universities
  - National Research University status since 2010
Centers and institutes

- Center of metallurgy and material study
- Center of mechanical engineering
- Nanotechnology research and education center
- Research-manufacturing institute "Educational engineering and technologies"
- Supercomputer Simulation Laboratory
Supercomputer Simulation Laboratory

http://supercomputer.susu.ac.ru/en/

- **Supercomputer Center**
  - supercomputers administration and software license management
  - research in parallel and distributed computing
  - development of software for grid computing and supercomputer systems

- **Distributed Computing and Embedded Systems Department**
  - development of software for distributed computing, embedded systems, mobile platforms, electronic resources

- **Support and Training Department**
  - consultation for the users of the applied and system software
  - training courses based on the SSL resources

- **Data Mining and Virtualization Department**
  - research in the field of data mining and virtualization technologies, solutions of practical problems based on these technologies, implementation and maintenance of appropriate software
Supercomputer Resources
Evolution in SUSU

**SKIF URAL**
Supercomputer

**SKIF-Aurora**
SUSU (upgraded)
Supercomputer

**Physics**
Computer cluster

**Infinity**
Computer cluster

Peak performance
1 Gigaflops 2000

**Peak performance**
333 Gigaflops 2004

**SKIF URAL**
Supercomputer

Peak performance
16 Teraflops 2008

**SKIF-Aurora**
SUSU
Supercomputer

Peak performance
24 Teraflops 2010

**SKIF-Aurora**
SUSU (upgraded)
Supercomputer

Peak performance
117 Teraflops 2011

**Intel Pentium 3**
0.8 GHz

**Intel Xeon 64**
3.2 GHz

**Intel Xeon E5472**
3.0 GHz

**Intel Xeon x5570**
2.93 GHz

**Intel Xeon x5680**
3.33 GHz

9-May-12 Database Systems Research Group Scientific Seminar
SKIF-Aurora SUSU Supercomputer

- Russian TOP50 list: 4th (Apr 2012)
- World TOP500 list: 121st (Nov 2011)
- Peak performance: 117 Teraflops
- Qty of computing cores: 8832
- RAM: 9 TByte
- Disk space: Intel SSD 108 TByte

Communication networks
- System network: 3D torus, 60 Gbit/s
- InfiniBand QDR, 40 Gbit/s
- Gigabit Ethernet
Omega project [http://omega.susu.ru](http://omega.susu.ru)

- **Goal**: development of a prototype of parallel RDBMS for cluster systems
  - Based on partitioned parallelism and EXCHANGE operator
  - Testbed of various research ideas (data replication, load balancing etc.)
- **Grants**: financially supported by the Russian Foundation for Basic Research
- **Outcome (since 1997)**
  - **Papers**: more than 60 and 10 papers in recognizable Russian and International scientific proceedings/journals, respectively
  - **Talks**: more than 50 and 10 talks at Russian and International scientific conferences (incl. ADBIS, DASFAA, DEXA), respectively
  - **Dissertations**: 1 Dr. of Science, 4 Cand. of Science
Partitioned parallelism

PARTITIONING by means of fragmentation function

Original relation

Resulting relation

MERGING
Serial vs parallel query plan

SELECT Name
FROM S
WHERE City='Heidelberg'

Serial plan

Parallel plan
Exchanges of tuples: not needed

- **P (PID, Name)** – parts
  - partitioned using SID
  - \( \varphi_S(t) = t.SID \mod N \)
- **S (SID, Name)** – suppliers
  - partitioned using SID
  - \( \varphi_{SP}(t) = t.SID \mod N \)
- **SP(SID, PID, Qty)** – supplies
  - partitioned using SID
  - \( \varphi_{SP}(t) = t.SID \mod N \)
- **SELECT Name**
  **FROM S, SP**
  **WHERE S.SID=SP.SID and S.City='Heidelberg'**

No exchanges needed.
Exchanges of tuples: needed

- P (PID, Name) – parts
- S (SID, Name) – suppliers
  - partitioned using SID
  - \( \varphi_S(t) = t.SID \mod N \)
- SP(SID, PID, Qty) – supplies
  - \textbf{partitioned using PID}
  - \( \varphi_{SP}(t) = t.PID \mod N \)
- SELECT Name
  FROM S, SP
  WHERE S.SID=SP.SID and
  S.City='Heidelberg'

Exchanges of SP\(_i\) 's tuples needed
EXCHANGE operator

- Exchange port $p$ means ID to differ such operators.
- Exchange function $\psi$ returns a number of node where tuple should be processed.
- Pseudo code

```java
if ($\psi(t) == \text{Mynode()}$)
    \text{Put}(t, \text{this\_output\_buffer});
else {
    \text{Send}(t, \psi(t));
    \text{Put}(t, \text{that\_output\_buffer});
}
```
Parallel agent

- P (PID, Name) – parts
- S (SID, Name) – suppliers
  - partitioned using SID
  - \( \varphi_S(t) = t.SID \mod N \)
- SP(SID, PID, Qty) – supplies
  - partitioned using PID
  - \( \varphi_{SP}(t) = t.PID \mod N \)
- SELECT Name
  FROM S, SP
  WHERE S.SID=SP.SID and
  S.City='Heidelberg'
Parallel agents

\[ \pi \text{Name} \]  
\[ \bowtie \text{SID} \]  
\[ \sigma \text{SP}_0 \]  
\[ \text{port}=1 \psi = \text{MERGER\_NODE} \]

\[ \pi \text{Name} \]  
\[ \bowtie \text{SID} \]  
\[ \sigma \text{SP}_{n-1} \]  
\[ \text{port}=1 \psi = \text{MERGER\_NODE} \]
EXCHANGE operator

merge

gather

split

scatter
PargreSQL project

- PargreSQL = PostgreSQL + Partitioned parallelism
DBMS processes: PargreSQL

par_Frontend
- user 1
queryexec - executor

connects (n)

Daemon (n) -> par_Backend (k)

<<create>> (1)

Backend
DBMS processes: PargreSQL
Query processing: PargreSQL

Parser
- parse

Rewriter
- rewrite

Planner
- plan/optimize

Parallelizer
- parallelize

Executor
- execute

Balancer
- balance
DBMS architecture: PargreSQL
Components deployment: PargreSQL

Server
par_Backend

Client
libpq-fe
par_libpq-fe

dbname=postgres
hostaddr=10.1.6.16
port=5432

dbname=postgres
hostaddr=10.1.7.1
port=5432

dbname=postgres
hostaddr=10.1.7.3
port=5432

dbname=postgres
hostaddr=10.1.7.4
port=5432

dbname=postgres
hostaddr=10.1.7.5
port=5432

dbname=postgres
hostaddr=10.1.7.6
port=5432
Migration of applications

PostgreSQL App

```c
// app.c
#include <libpq-fe.h>

void main()
{
    PGconn c = PQconnectdb(...);
    PResult r = PQexec(c, ...);
    ...
    PQfinish(c);
}
```

PargreSQL App

```c
#include <par_libpq-fe.h>

void main()
{
    PGconn c = PQconnectdb(...);
    PResult r = PQexec(c, ...);
    ...
    PQfinish(c);
}
```

```
define PQconnectdb(...) \par_PQconnectdb(...) 
define PGconn \par_PGconn ...
```
PargreSQL: partitioning

- create table S (  
  SID integer primary key,  
  Name char(50))  
with (fragattr = SID);  
-- Set SID as fragmentation attribute  
-- with fragmentation function SID % N,  
-- where number of nodes N  
-- is a number of lines in par_libpq.conf file.
PargreSQL: Parallelizer
**PargreSQL: Parallelizer (INSERT)**

- `insert into T values (...)`;
- `insert into T select ...`;

Diagram:

```
Result

filter(t.fragattr % n = mynode)
```

```
E

port=0 \psi \equiv t.fragattr \% n

... ...
```

...
PargreSQL: Parallelizer (UPDATE)

Exchange behaves differently:

if (IsFoe(t)) {
    dup=Duplicate(t);
    t.SystemFlag=DO_INSERT;
    dup.SystemFlag=DO_DELETE;
    Send(dup, \psi(t));
    return (dup);
} else {
    do as usual
}
PargreSQL: Message Passing

- **Why not plain MPI?**
  - Because of a `fork()` inside the PostgreSQL daemon
  - Use shared memory for exchanges within one node, otherwise MPI

- **MPI-like interface**
  - `Init()`
  - `Finalize()`
  - `GetRank()`
  - `GetSize()`
  - `IRecv()`
  - `ISend()`
  - `Test()`
  - `Finalize()`
Current results

- **Implemented**
- **To Do**

**Source code size:** 5K lines
Experiments

- **Hardware**
  - SKIF-Aurora SUSU
  - Nodes: 1 to 10

- **Database (synth. data)**
  - T1 \( (f_0, f_1) \), frag attr is \( f_1 \), \( 10^9 \) tuples
  - T2 \( (f_0, f_1) \), frag attr is \( f_1 \), \( 10^5 \) tuples

- **Queries**
  - select * from T1, T2 where T1.f1=T2.f1
  - select * from T1, T2 where T1.f1=T2.f0
  - select * from T1, T2 where T1.f0=T2.f1
  - select * from T1, T2 where T1.f0=T2.f0
Experiments

![Graph showing speedup vs nodes for different scenarios. There are four scenarios labeled (1) to (4). Each scenario is represented by a different symbol on the graph.](image-url)
Related work: ParGRES

- ParGRES is a middleware over cluster of PostgreSQL DBMSes to process OLAP queries.

Related work: GParGRES

- **GParGRES** is an extension of ParGRES for grids.

Team

- Leonid Sokolinsky
  - Prof., Dr. of Science (Phys&Math)
  - Dean of Computational Mathematics and Informatics, Head of SSL

- Mikhail Zymbler
  - Assoc. Prof., Cand. of Science (Phys&Math)
  - Head of DM Dept, SSL

- Constantin Pan
  - Postgraduate student
  - Programmer of DM Dept, SSL

- Ruslan Miniakhmetov
  - Postgraduate student
  - Programmer of DM Dept, SSL
Team (continued)

- **Aleksey Koltakov**
  - Master student

- **Elena Aksenova**
  - Postgraduate student
  - Programmer of the S&T Dept, SSL

- **Lyudmila Utkina**
  - Master student
  - Programmer of the S&T Dept, SSL

- **Alexander Medvedev**
  - Master student

- **Evgeny Gavrish**
  - Master student
Thank you for paying attention!

- Questions?
  - Mikhail Zymbler
    zymbler@gmail.com
  - Constantin Pan
    kvapen@gmail.com

- More info
  - http://omega.susu.ru