Norwegian
Meteorological Institute
met.no

The Yr.no Backend System

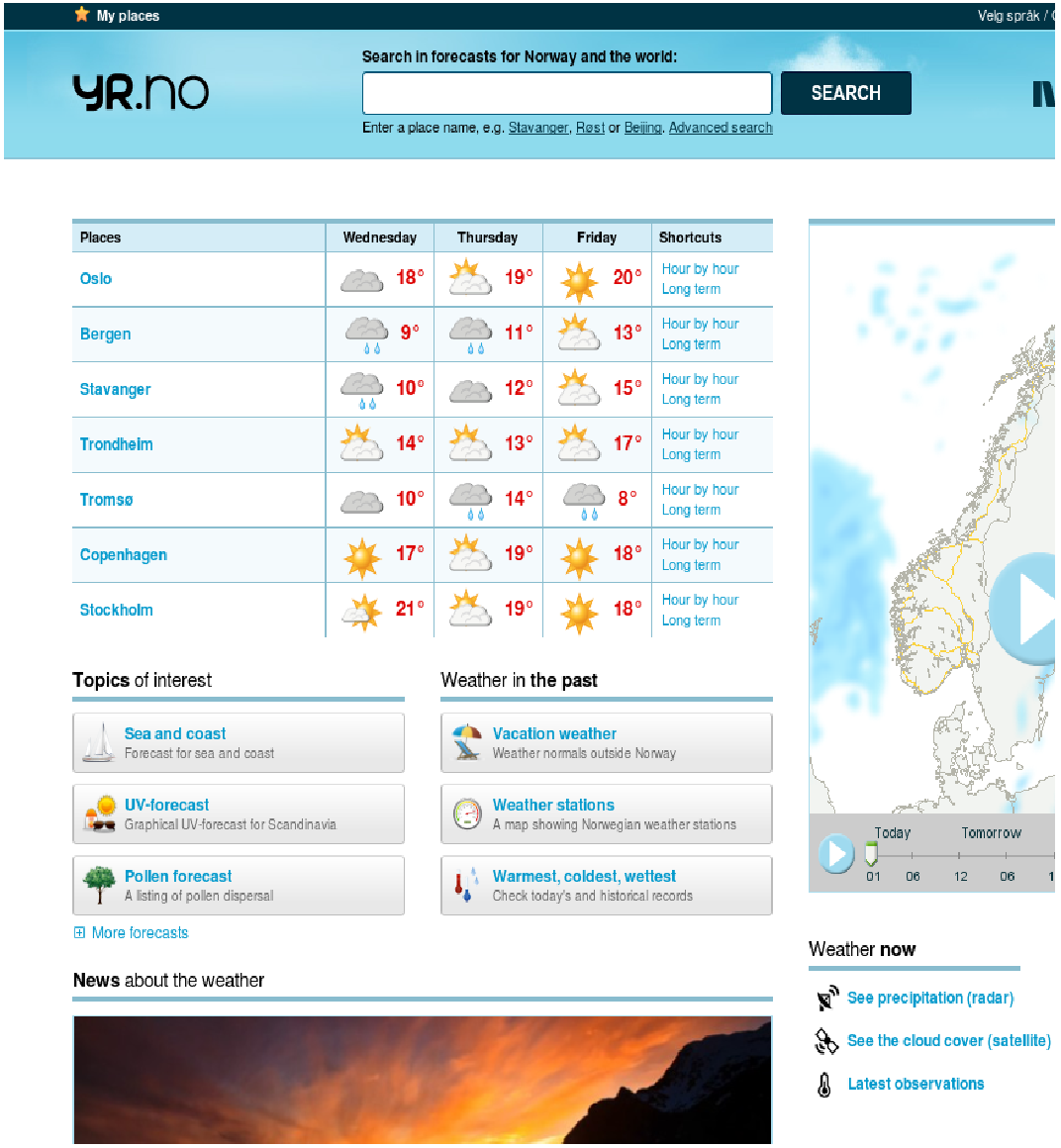
EGOWS 2011, Toulouse, France.
June 6 - 9, 2011

Who am I?

- Michael Akinde
  - Database Architect, Met.no (2006 –)
  - Database Specialist, SMHI (2002 – 2007)
  - Ph.D. Computer Science, Aalborg University (2003)
- WDB + WMS Development Group
  - Data storage and transfer backend of Yr.no
  - Four software developers
  - Two to three operational developers (setup, maintenance)

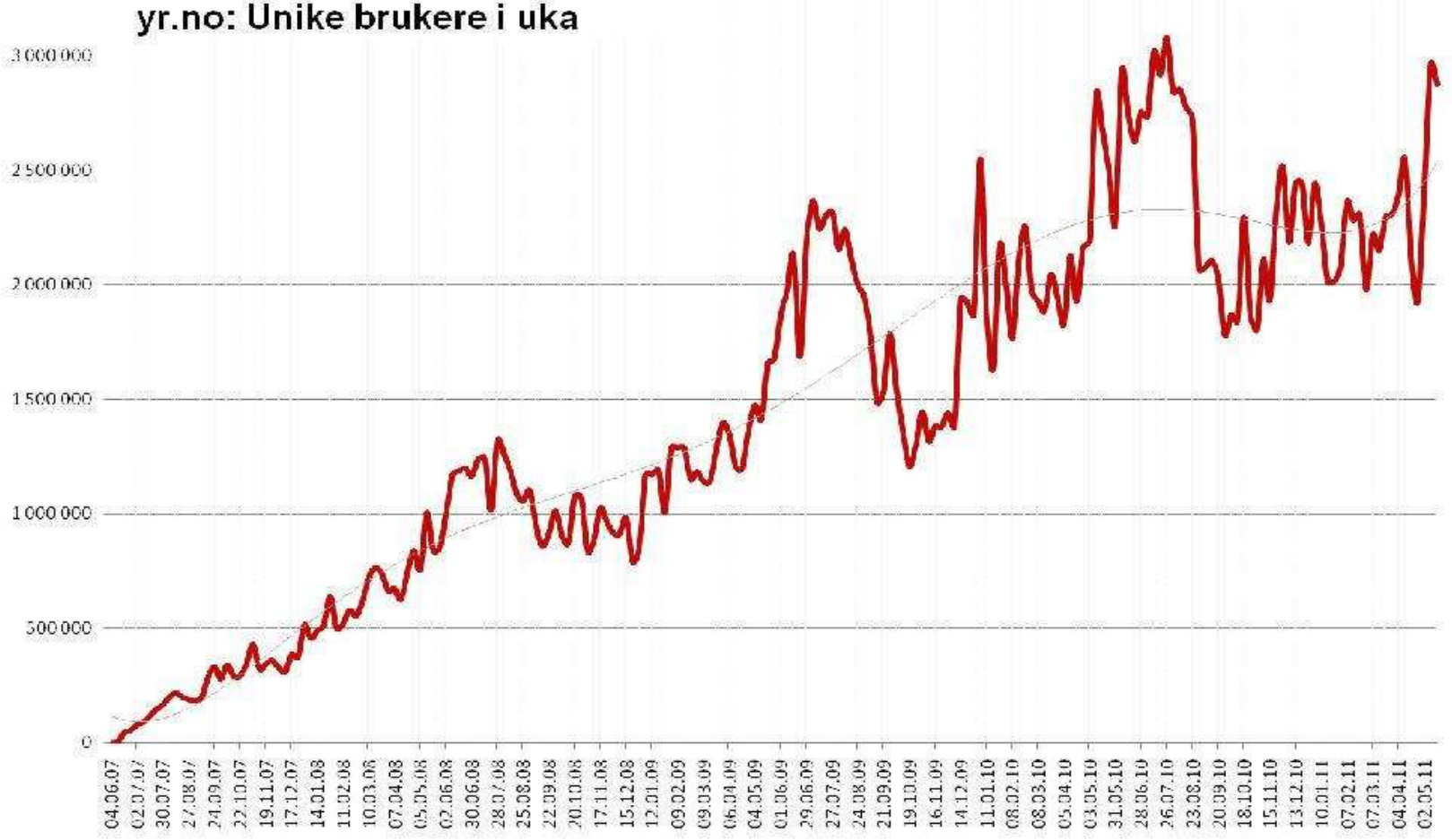
Yr.no

- Launched 19-9-2007
- One of Norway's largest web sites
  - ~2,2 million unique clients every week
- “All” Norwegian internet users know of yr.no
- More than a million Norwegians use yr.no every day



Yr.no Statistics

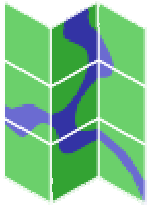
- 55% Norwegians, 27% Swedish; the last 18% is distributed over other countries



Yr.no Backend: The Goals

- 24x7 hour operation
- Bursts of data input
  - ~80 GB of data 4 times a day
- Peaks of data output
  - ~120,000 data values retrieved per second
- Sub-second response times (300 ms)
- Scalable
- Flexible

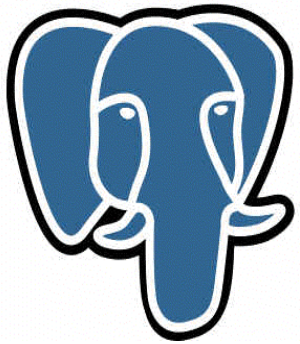
How did we do it?

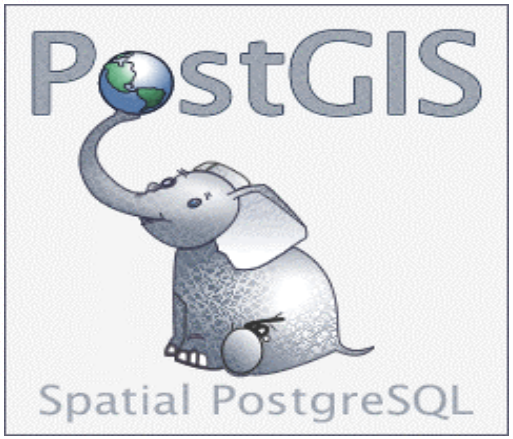


MapServer
open source web mapping

boost
C++ LIBRARIES

PostgreSQL





PROJ.4

Varnish

OpenLayers

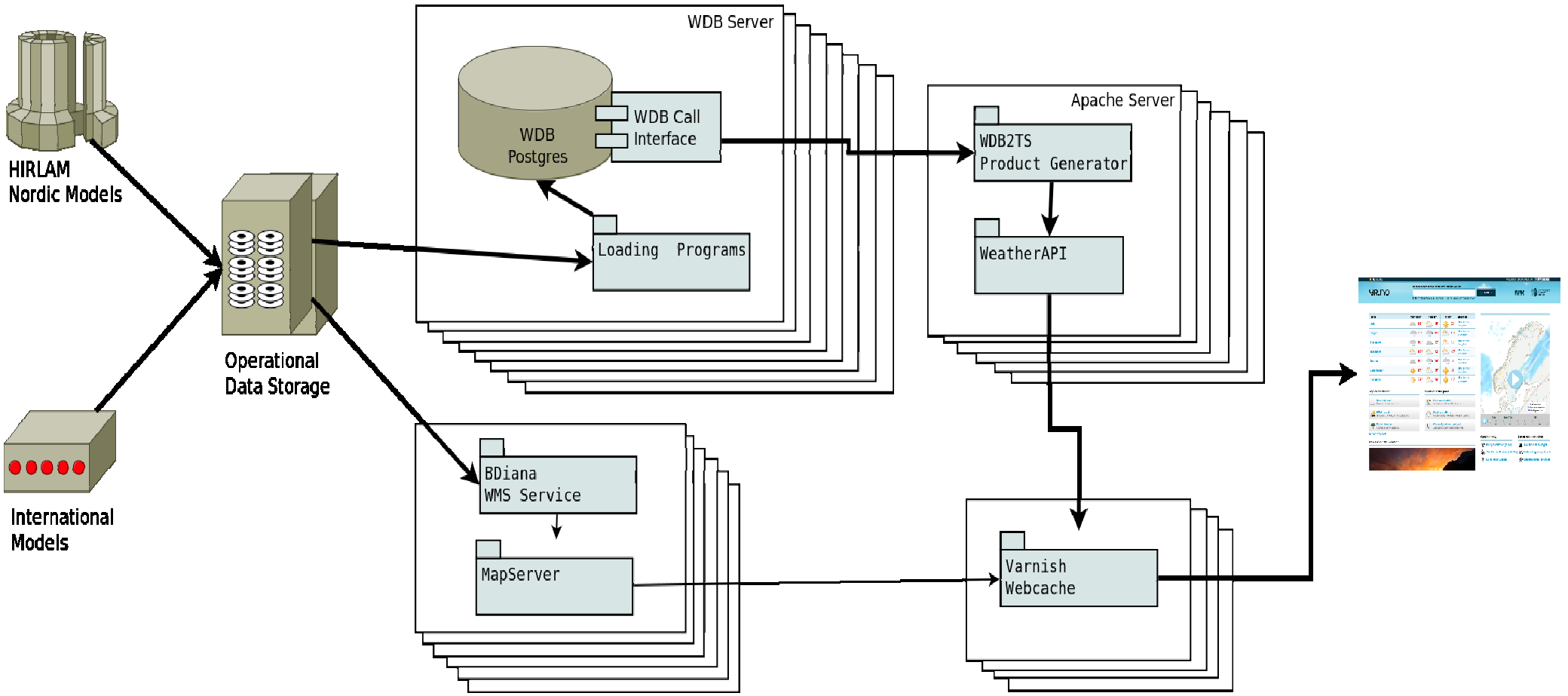
APACHE

What went into the Backend?

- Storage Backend: Postgres database system for storage and retrieval of weather data (WDB)
- Data Retrieval: Query interface through Apache module (WDB2TS)
- Image Generation: Diana batch image generation (BDiana)
- Cache: Varnish web-cache

- 100% Free and Open Source

Yr.no Backend Architecture



Yr.no Backend Details (1)

- Operational Data Storage
  - 2 Distributed Replicated Block Device NFS Servers
- Weather and Water Database System
  - 6 servers for weather data; 2 for ocean data
  - Each server delivers 20-40 requests per second with ~0,25 seconds latency
  - A typical request retrieves/calculates ~1500 floating points from ~80Gb of data

Yr.no Backend Details (2)

- Product Generation Servers
  - 6 servers running Apache HTTP Server
  - 400-600 data requests/second on average; peak loads at ~1000 data requests/second
- Image Generation Servers (WMS)
  - 1000-1500 image requests/second on average; peak loads at ~2000 image requests/second
- Cache Servers (Varnish)
  - 2 active servers + 2 standby servers. Delivers >1000 requests/second with <0.1 seconds latency

WDB: Database Backend

- WDB is a storage system for meteorological, hydrological, and oceanographic data
  - Open-source project (GNU GPL2) developed and currently administrated by met.no
  - Built on open source components
    - Postgres Relational DBMS
    - PostGIS, Proj.4, libgeos
    - GNU tools, Boost, log4cpp

WDB: Loading Programs

- Loading programs write data into the WDB system using write functions in the WDB Call Interface
  - GRIB1/GRIB2 data (ECMWF GRIB API)
  - FELT data (internal met.no)
  - MOX data (XML format for weatherapi.met.no)
  - BIL (limited loader for Binary Interleaved by Line)
  - DEM (Norwegian demographic format)
  - Pending: NetCDF, BUFR

WDB: Call Interface

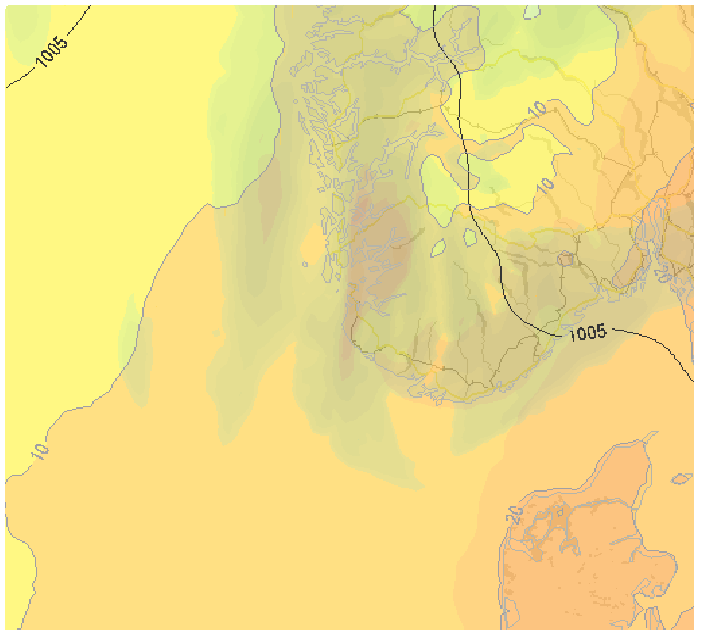
- WCI SQL Function Interface
  - wci.read(…)
    - Returns data as grid reference or point data
    - Slice data by data provider, time, parameter, levels, etc.
  - wci.write(…)
    - Writes data into the database; pretty effective for fields, not optimized for point data (yet)
    - Used by the WDB loading programs
  - Results of the SQL functions can be manipulated using regular SQL and accessed from any language

WDB2TS

- Simple web front-end for retrieving point-data from a WDB system
  - Apache module
  - REST-like interface to the data
  - Returns data in two XML formats (weatherapi legacy format and MOX) as well as comma separated text
- MOX is a GML application schema developed at met.no for representing time series in XML

Image Generation

- Builds on the Diana meteorological visualization and production software developed at met.no
  - Visualization of fields, satelitte and radar images, surface observations, weather charts, etc.
- OpenLayers
- MapServer



WDB: What are the problems?

- Common Claims:
  - Relational Databases are not suitable – they do not support multi-dimensional arrays
  - Poor performance on large arrays
  - Databases are general purpose and there is therefore a significant overhead in both resources and performance

Are they problems?

- The advantage of extending a relational database system:
  - You build the functionality that does not exist, or is not handled well in the RDBMS
  - You get huge amounts of functionality for free
  - When you need to add new functionality, you rarely need to build from scratch
  - You build on an extremely well-tested base
    - E.g., modified version of Postgres are used by Yahoo!, reddit, Skype, ISS (International Space Station), etc.

But...

- Performance
  - You can always do things faster in some other way, if you spend enough time on it
- Space constraints
  - WDB works with uncompressed data; it is built for speed, not saving space
- Databases are general-purpose!
  - Yes – that does mean a lot of functionality that one will (maybe) never need

So why use the DBMS?

- Some of the stuff you get for free:
  - Data-independent API
  - Near universal platform support
  - SQL Query > complex routines
  - Easy to browse
  - Transactions, multiple user handling
  - Scalability
- A flexible, efficient, robust and sharable system architecture

What was the point again?

- High Performance
  - Response times from ~80ms (main memory) to ~240ms (database)
- Scalable
  - More performance ► New server
- Flexible
  - Able to store many types of meteorological and oceanographic data (point data, grids, binary data)
  - Delivers data in various forms (data points or binary streams)

- Robust storage system that can be leveraged

Future Plans

- Additional data support
  - Integration with NetCDF-Java
- Extreme scalability
  - Distributed database technology
  - Distributed memory caching
- Data compression
- Point data optimization
- Fixing bugs and adding/improving features

Open-source

- WDB System
  - On sourceforge: http://wdb.sf.net
  - Source code, subversion repository, documentation (wiki and manuals)
  - Announcements on freshmeat.net
- Diana
  - Check http://diana.met.no

Why is it open-source?

- We want others to use it, of course!
  - Reliability and Stability are very important. More users = more bugs found = more bugs fixed
  - The code is open = if you find a bug and we don't have the time to fix it, do it yourself
- Benefit the international meteorological community
- Open-source software is a key element in the Norwegian Government's ICT strategy

Conclusions

- Yr.no is built on an entire tool chain of free and open source systems
  - Reliable
  - Flexible
  - Efficient
- Open source software exist to try, test, and tweak.
  - You're welcome