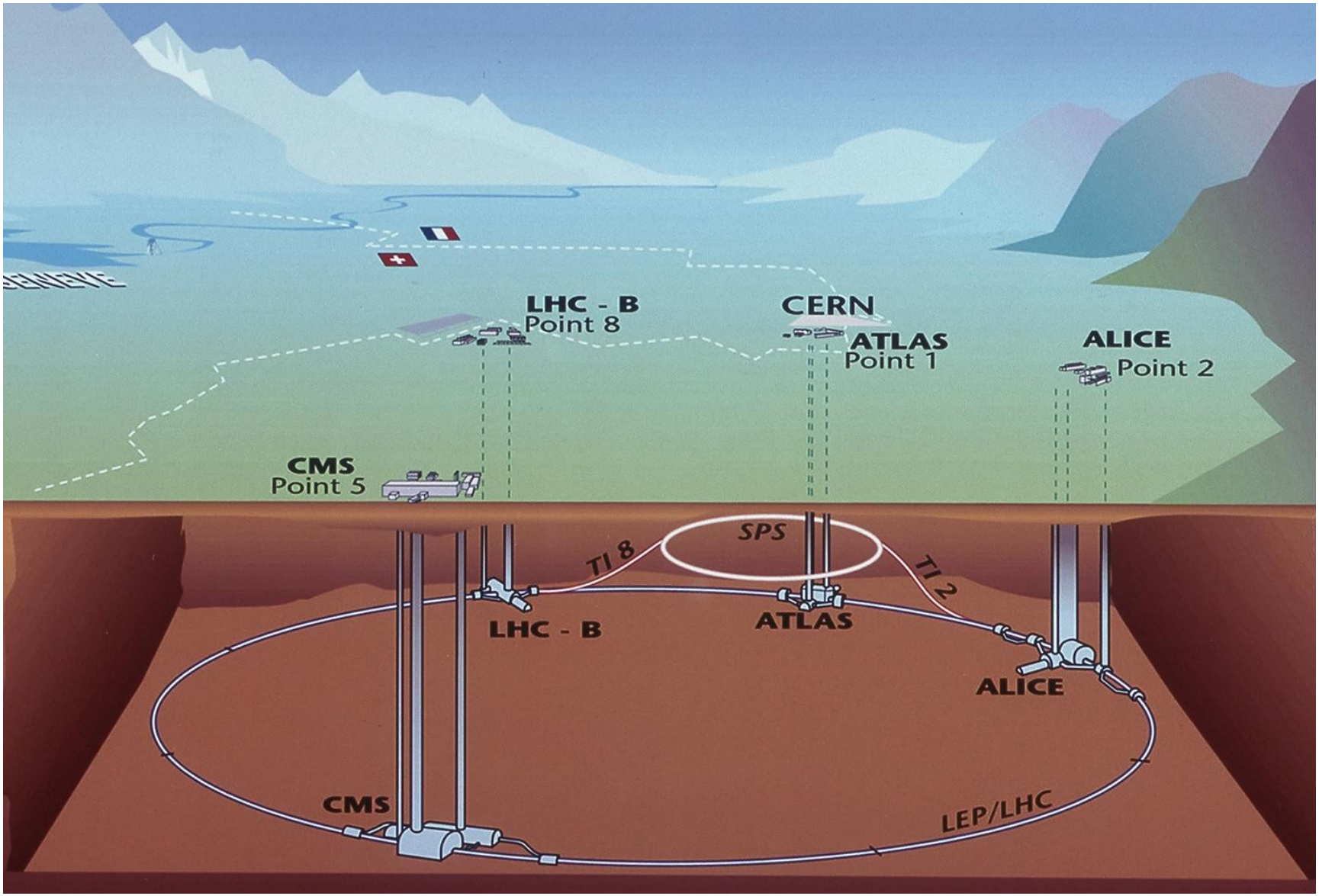
Grid usage by LHC/CMS

Derek Feichtinger, PSI

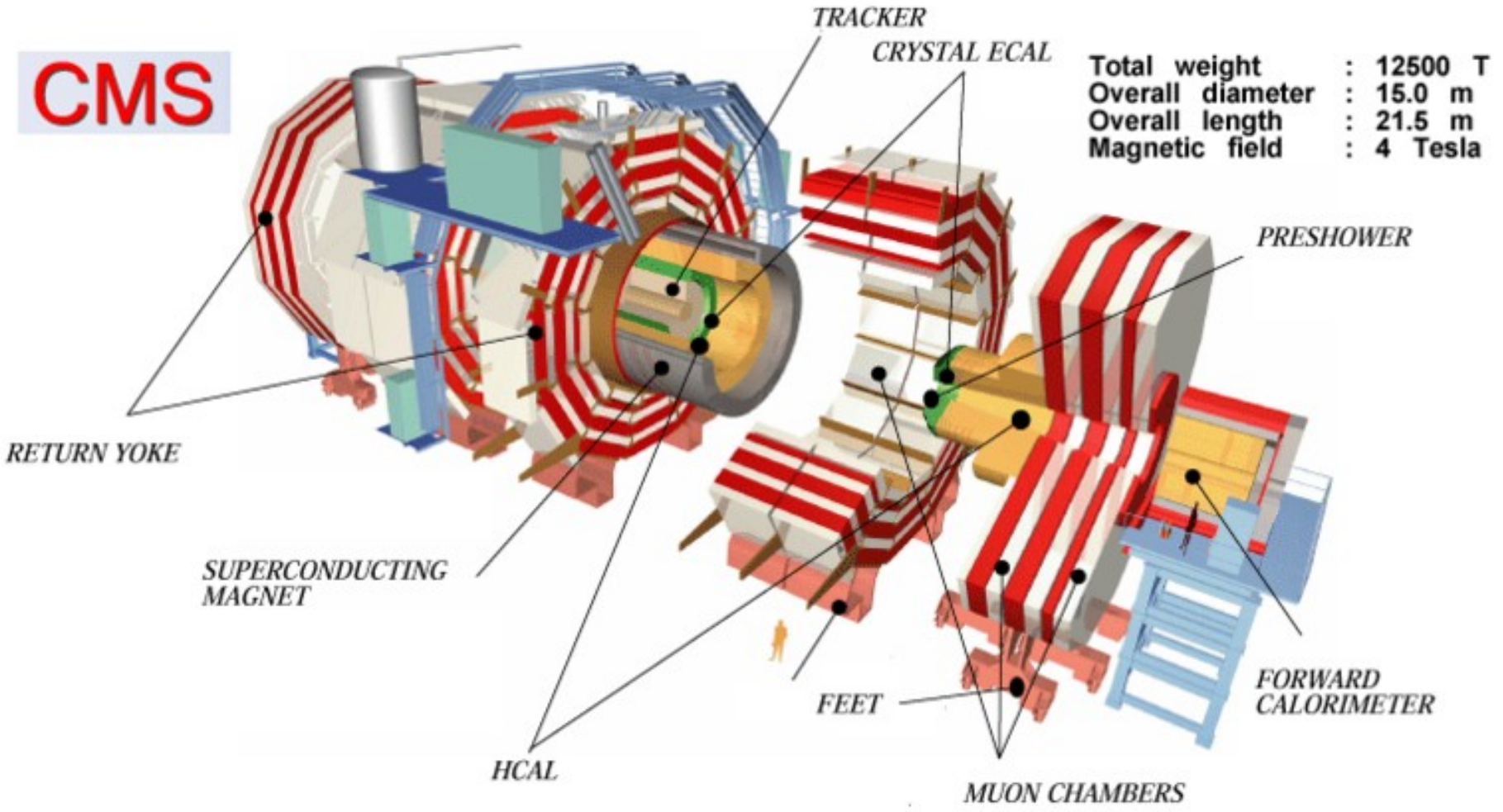
Contents

- Typical HEP job types
  - CMS data model
- Why use a Grid?
- A hierarchically organized Grid
  - Data flows and data transfer system
  - An example CMS workflow
  - Software Management
- Efficient file transfers
- Experiences

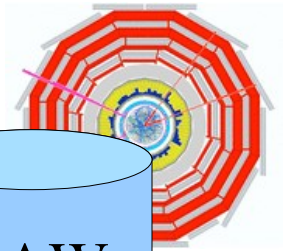
CERN / LHC


LHC - B
Point 8
CERN
ATLAS
Point 1
ALICE
Point 2
CMS
Point 5
SPS
TI 8
TI 2
LHC - B
ATLAS
ALICE
CMS
LEP/LHC

The Compact Muon Solenoid Detector



Typical Job Types

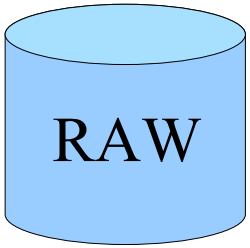


Monte Carlo Simulations

CPU bound

RAW

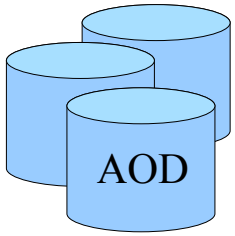
Reconstruction



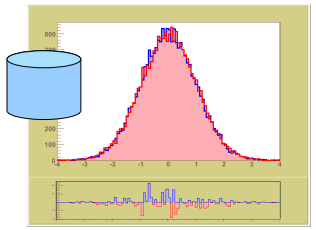
CPU bound

RECO/ AOD

Analysis



CPU or I/O bound



Jobs involve looping over large numbers of **independent** measurement data ("events") ➡ **Trivially parallelizable, ideal for Grid**

But: jobs need large amounts of data which must be available. Required I/O bandwidth is moderate (~ a few MB/s) for most jobs

CMS Data Model

2006 assumption for luminosity in 2008:

- $2 \cdot 10^{33}$ $cm^{-2}s^{-1}$ (10^7 s/year, 30% uptime)
- Writing of DAQ-RAW events at 150 Hz

Data Formats and Volumes

Format	Event Size /MB	Events / year	Data/year /PB
RAW	1.5	3.30E+09	5
RECO	0.25	8.30E+09	2.1
AOD	0.05	5.30E+10	2.6

Typical files contain large numbers of events.
Files usually are dimensioned to sizes of 1-2 GB.

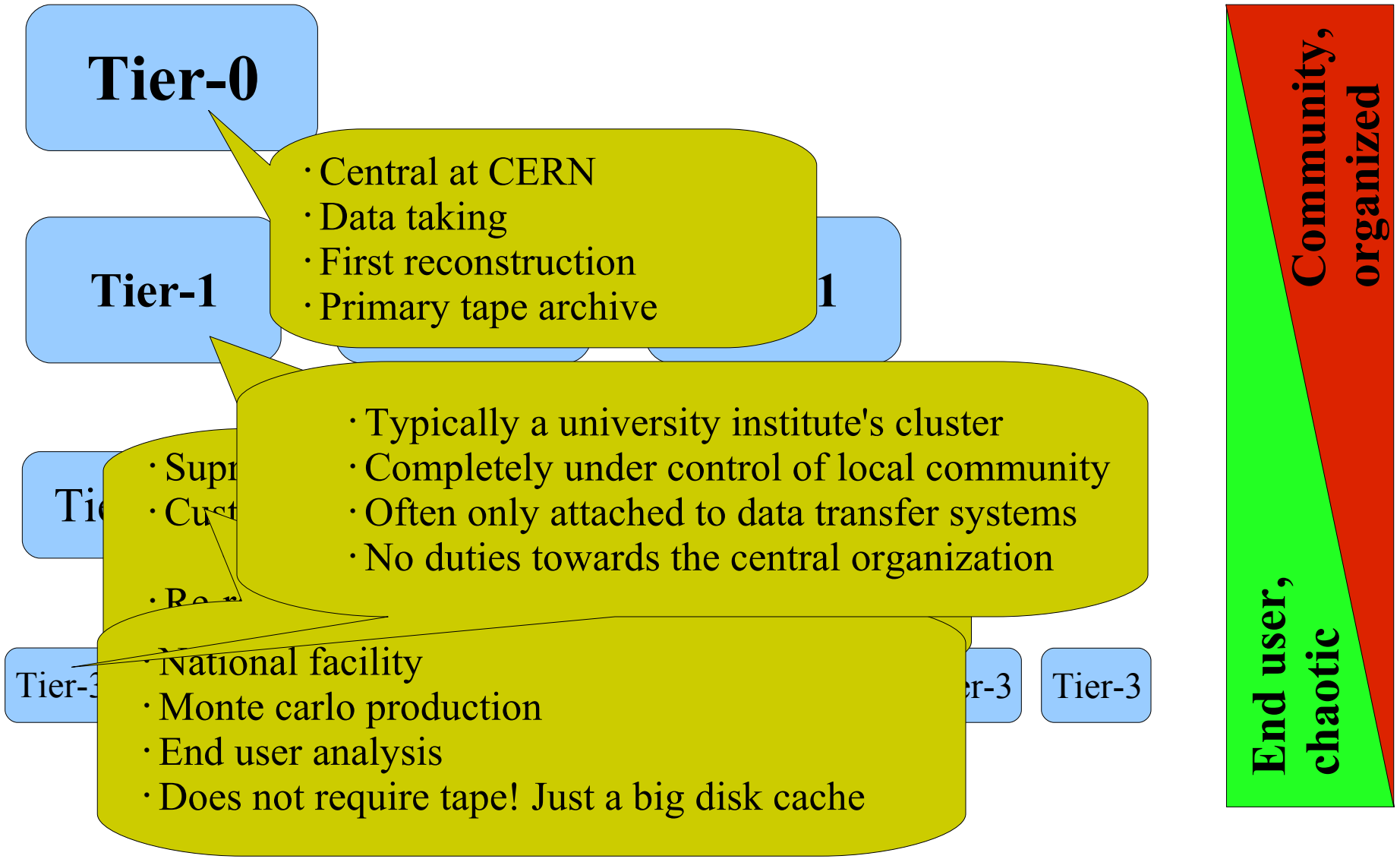
CMS Computing TDR:
http://cmsdoc.cern.ch/cms/cpt/tdr/index.html

Why use a Grid?

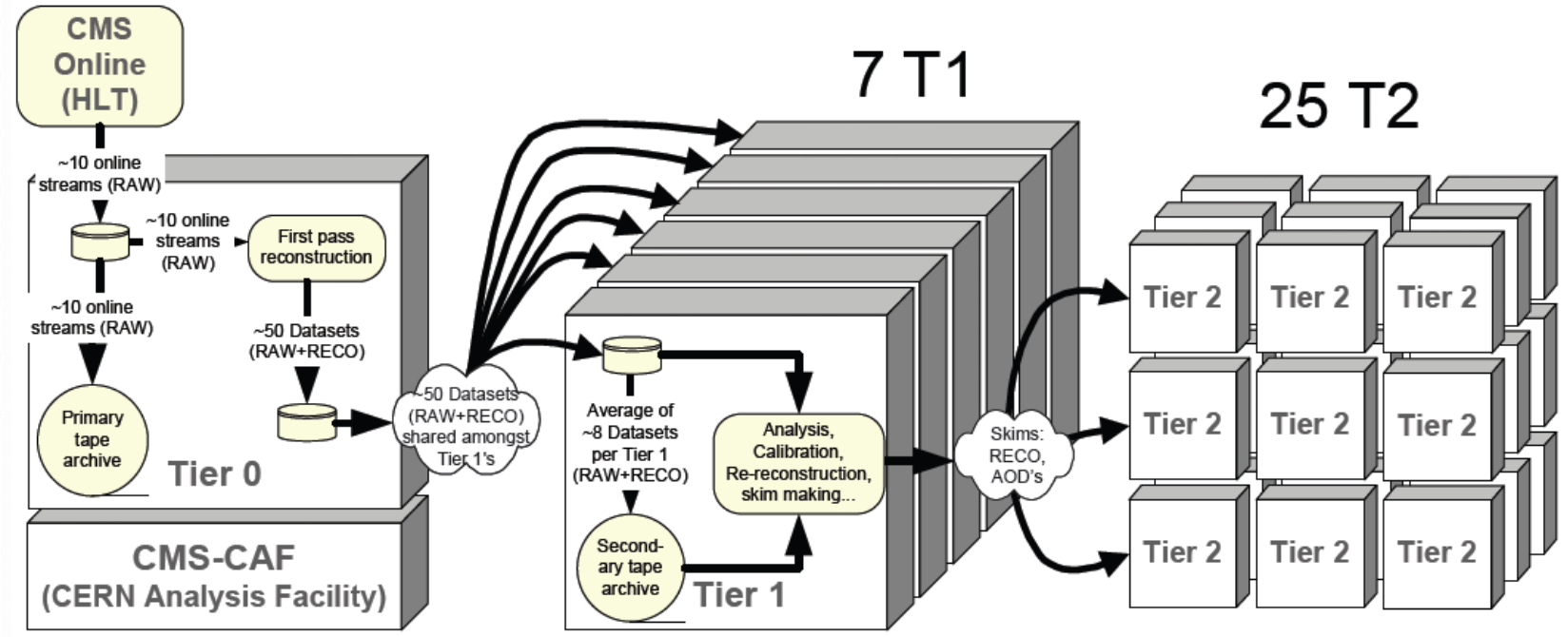
- Massive amount of data to analyze (> 10 PB/year)
- Building a huge central computing center at CERN to serve all the community is not possible
  - Member states want to invest in their own infrastructure
- Still, data must be accessible for all members
  - Cannot make copies of everything at every center, huge waste of resources. Need an organized data distribution.
  - Cannot retrieve remote data “just in time” into active jobs. Latency far too large for the amounts of data needed.
  - But if certain parts of the data only exists at some remote sites, need to enable users to run jobs there.
- Need a system that **sends jobs to sites hosting the data**

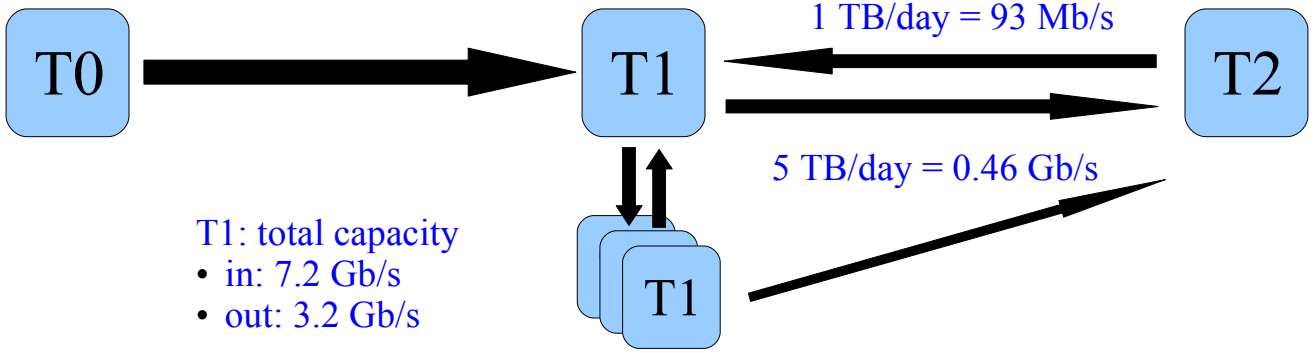
enter ***the Grid***

A hierarchically organized Grid

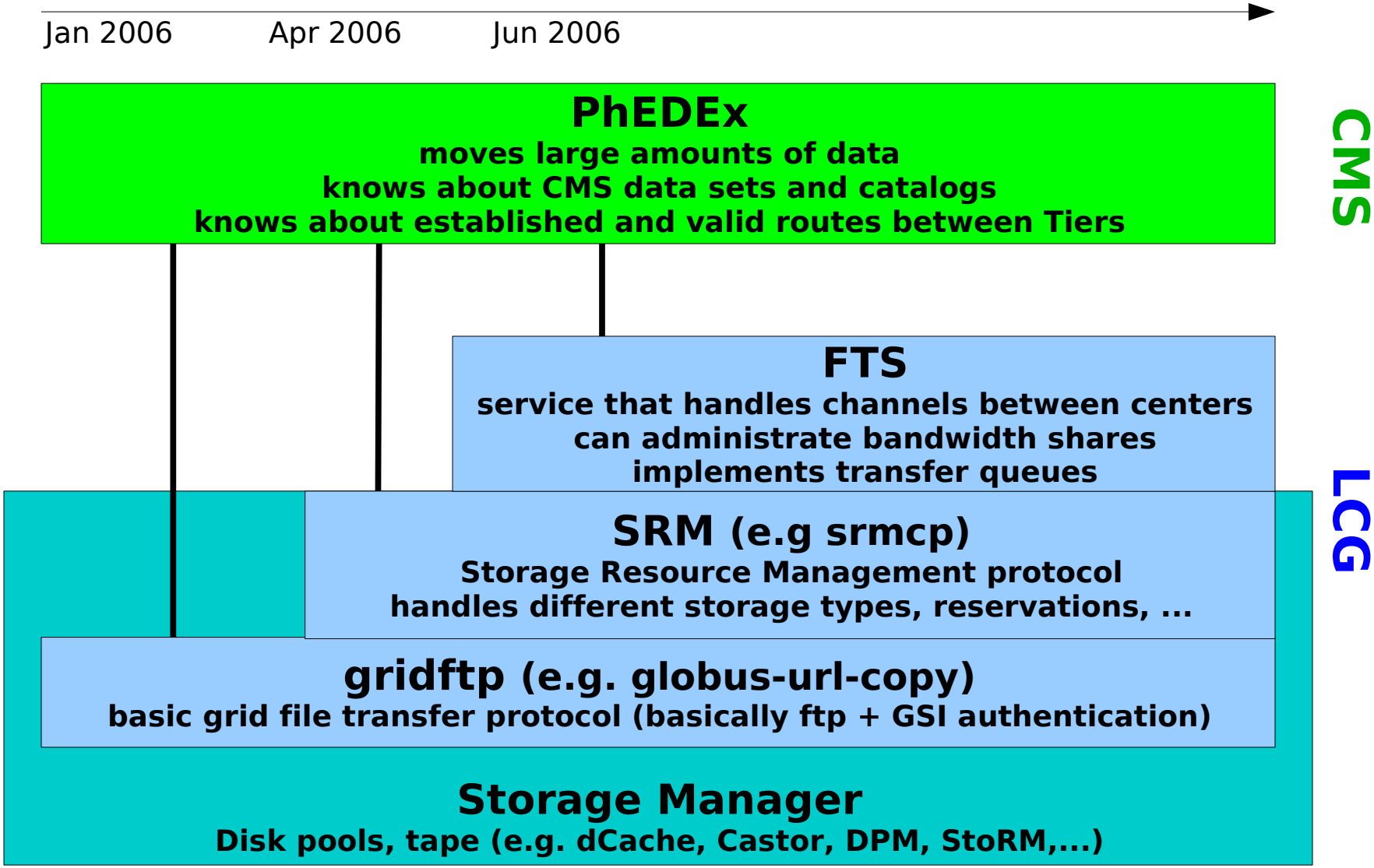


Data flows: Inter Tier





CMS Data Transfer System



Software Management

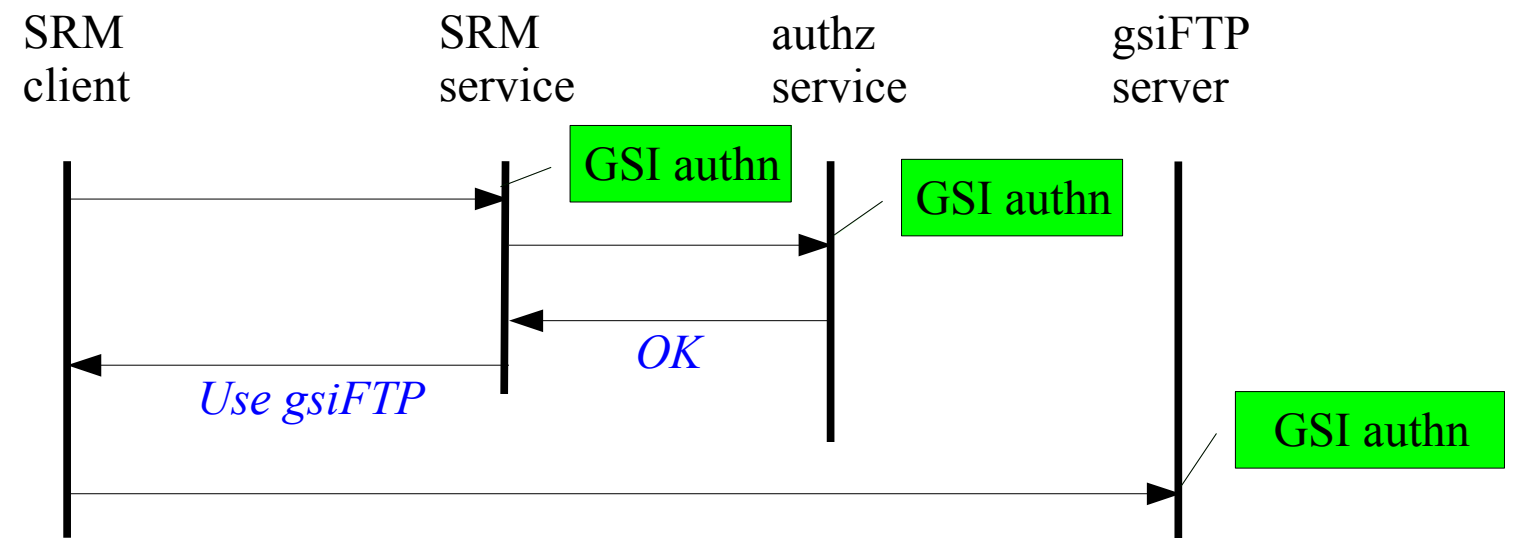
- User software frameworks are usually installed by central teams
  - Install jobs running as special VO user with write access to SW area
- The WLCG Grid is very homogeneous
  - Scientific Linux 4 is the standard OS → simpler SW deployment
  - Jobs find standard Grid environments based on environment variables
- CMS software packages
  - are managed by a user space RPM data base
  - Bring most of their dependencies (even glibc, python, etc.) - huge!
- SW Frameworks can be extended by users through plugins that are carried inside their jobs
- Test jobs from central teams regularly test the needed basic versions

An example workflow on the GRID

- Search a data set (DBS weblink)
  - Decide whether to stage it to some sites (PhEDEx order, manage requests)
  - Get configuration file templates
- Create a CMS framework work area on a local Grid-UI
  - You may test your jobs and plugins locally with a subset of data
- Configure a grid job
  - Usually a job generator framework (e.g. CMS CRAB) is used to convert the job into adequately sized sub jobs (*trivial parallelism*)
  - Define target SE area, whether to publish results to a DB, etc
- Submit
- Wait.... and monitor
- Retrieve Output
  - Resubmit any sub jobs which show an error state

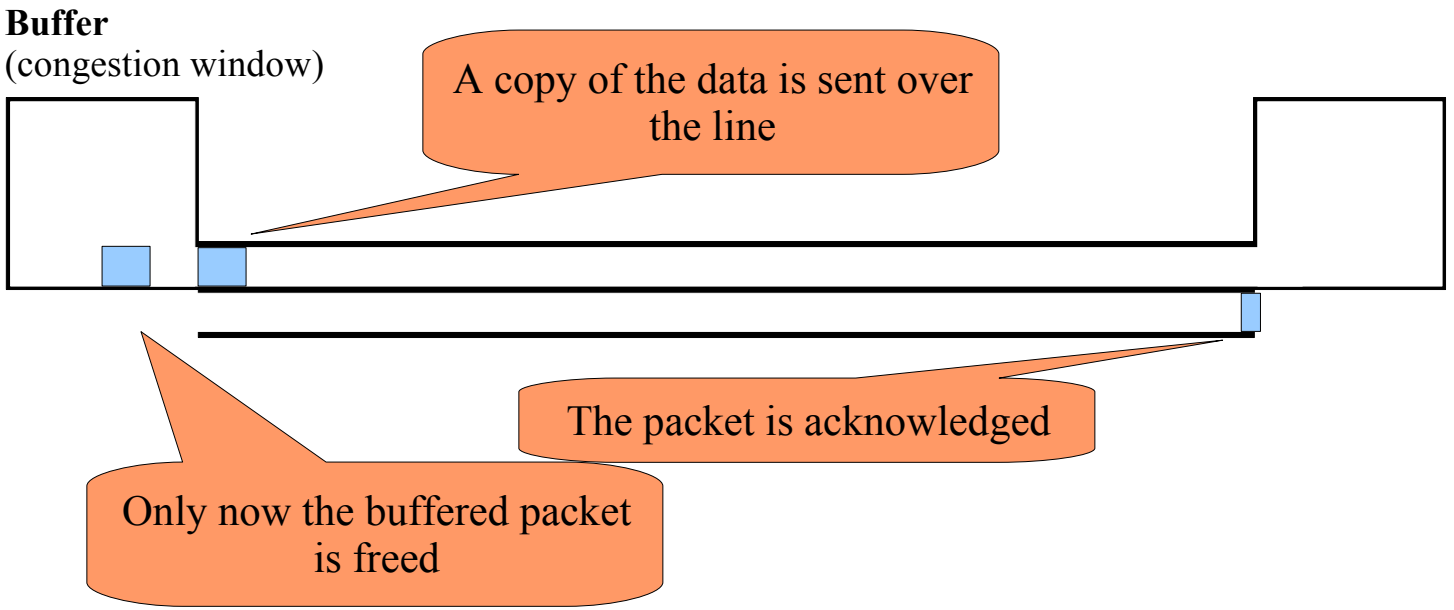
Efficient File Transfers

- Authentication through Grid certificates is expensive and can take fractions of a second
- Many operations involve several services and connections to be established



Use big files to minimize time spent shaking hands

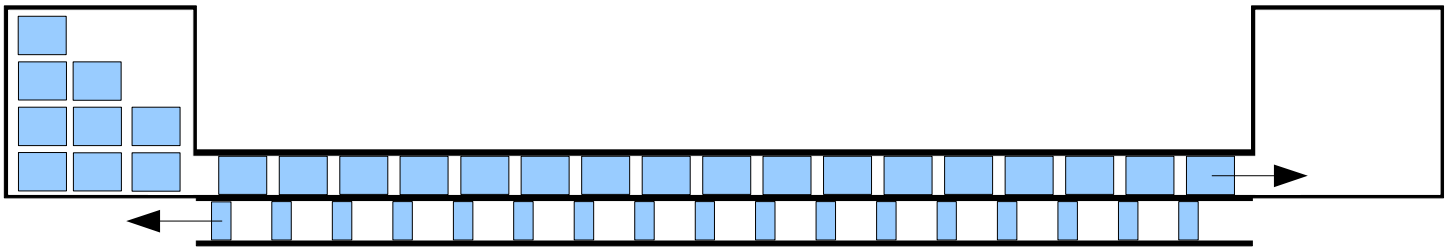
Background information: TCP packet transfer



- The time from the sending of the packet until the reception of the acknowledgement is called the *round trip time* (**RTT**)
- The number of packets that can be unacknowledged before transmission is paused, is increased with successful transfers until a threshold determined by the sender (*congestion window*) is reached

Efficient WAN Data Transfers

For an ideal TCP based connection...



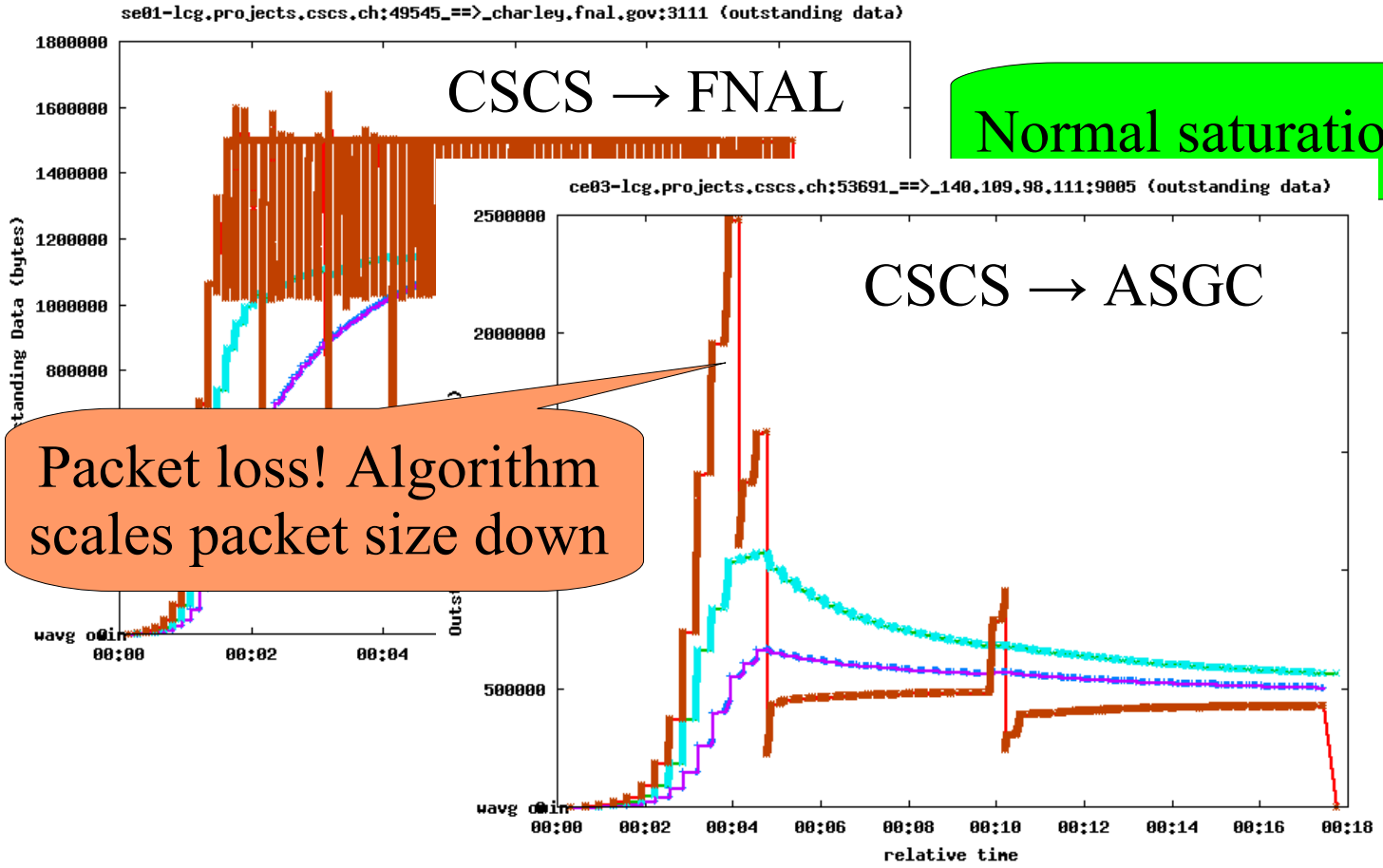
- The send buffer must be able to accommodate all packets not yet acknowledged
- The maximal volume of unacknowledged data is described by the *bandwidth delay product*

$$BDP = bw \cdot RTT$$

Example: T1_TW_ASGC to T2_CH_CSCS: $BDP = 1\text{Gb}/s \cdot 315\text{ms} = 0.315\text{Gb} \approx 40\text{MB}$

Serving many connection requires substantial memory (lowmem!)

Efficient WAN Data Transfers



- Try to minimize relative time spent in startup phase → use big files
- Try to find reasons for packet losses → network experts needed

Grid experiences: the price to pay

- All components need to deal competently with errors
  - Misconfigurations, temporary network problems
  - Make sure that it's not the weakest link that defines the chain: black hole effect
- Error propagation through the many software layers
  - Users often confronted with meaningless errors
- Service debugging highly non-trivial
  - Many tests need coordination between multiple individuals/sites
  - A site admin may not even be able to fully test his own site, because foreign VO credentials might be needed for some tests
- Su[illegible] of standards
  - Particularly [illegible] storage managers (e.g. through SRM)
- Users need to adapt to a different working style
  - Certificate handling, job submission, data management

Slow acceptance in user community

[illegible]re essential

Grid experiences: What to observe

- Intensive **interaction between developers and users**
  - Requirement documents are not enough
  - Need a sufficiently big test bed with real test users and use cases
  - Do *data challenges*, i.e. intensive periods of testing with set goals and where all sites are involved
- Try to follow (or establish) **standards** early on
- Services should keep understandable log files
  - Log levels/categories should be settable at runtime
  - There's enough packages for this around (e.g. *log4j* and friends)
- **Error passing** is an essential component
- **Keep it simple** – it's complicated enough
  - It would be nice to satisfy all requirements with esthetically perfect software design, but need to be pragmatic and advance in small steps and learn

Thank you

Questions?

Merry Christmas!

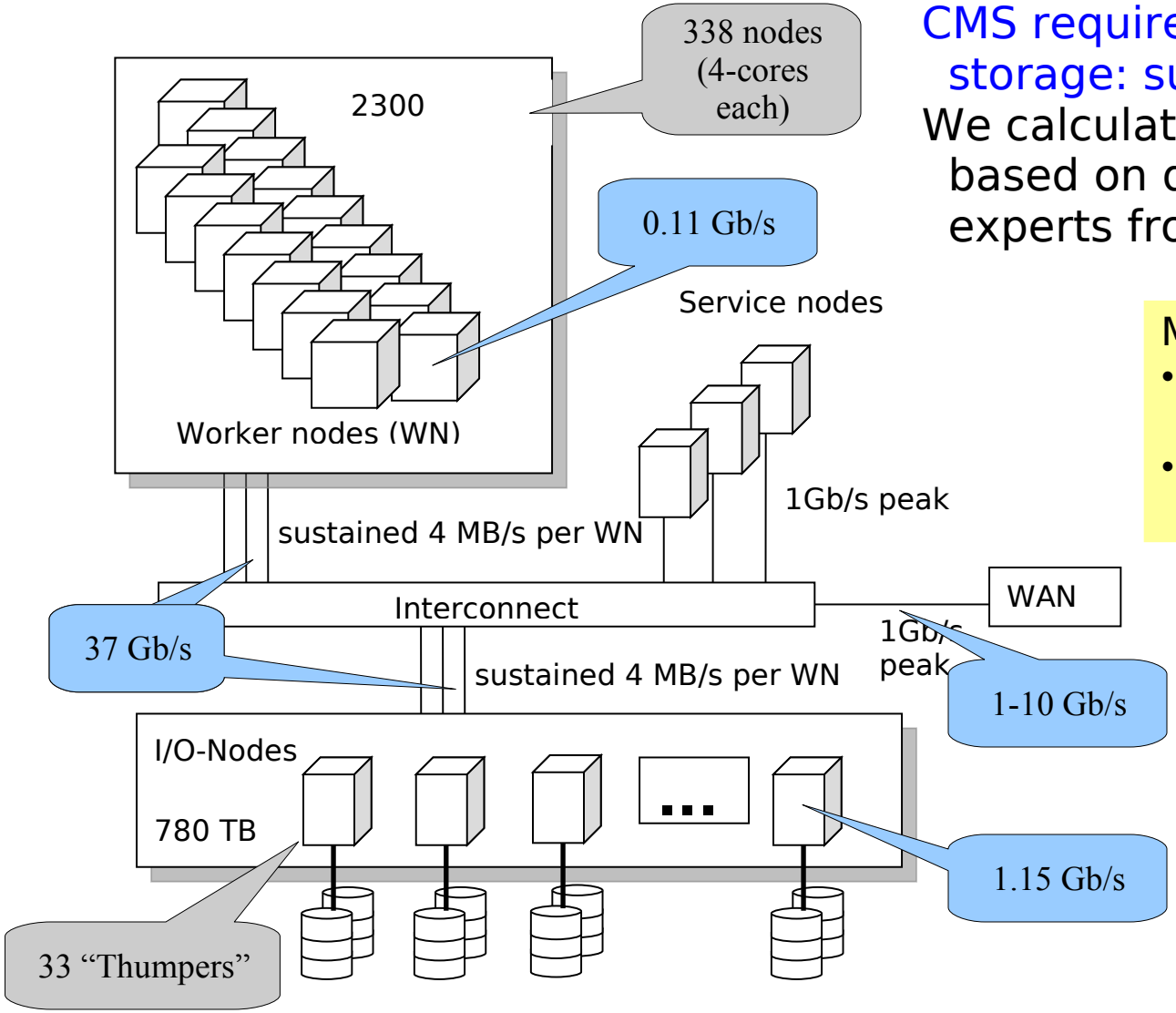
Backup Slides

EGEE in Numbers

- Number of sites connected to the EGEE infrastructure: 259
- Number of countries connected to the EGEE infrastructure: 52
- Number of CPUs available to users 24/7: ~ 72,000
- Storage capacity available: ~ 20 PB disk + tape MSS
- Number of registered Virtual Organisations: >130
- Number of registered users: >7500
- Number of jobs: >150k jobs/day

Data Flows: Intra-Tier2

Complete Swiss T2 in 2008:



CMS required bw from WN to storage: sustained 4MB/(s*kSI2k)

We calculate half of this value based on discussions with experts from other Tiers.

Model numbers for
- WN: SunFireX4100 (2*Opt 280), ca 1.7 kSI2k/core
- Storage: Sun "Thumper", 24 TB/node

Note: 1-Gigabit technology is sufficient for all nodes!