Scientific computing with Debian

mini-DebConf Paris

30 October 2010

LEADING THE ENERGY CHANGE

Context

Heavy scientific computing use at EDF

Why we use Debian
What we can achieve with it
What challenges it involves

EDF, a world energy leader

37,9 **millions**
of **clients** worldwide

169 139
collaborators worldwide

66,3 **G€**
revenue
49 % outside of France

618,5 **TWh**
produced worldwide

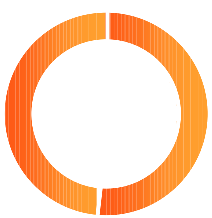
117,1 **g**
CO_2 per produced kWh

Key figures

- World leader in nuclear power operation, Europe leader in hydro power
- **Strong Europe implantation**: France, UK, Germany, Italy…
- Europe leader in distribution, transport and sales
- Industrial operator in **Asia** and **United States**
- Natural gas : a major player

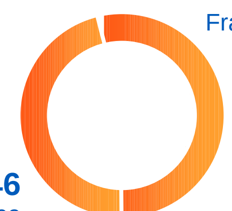
Revenue 2009

%



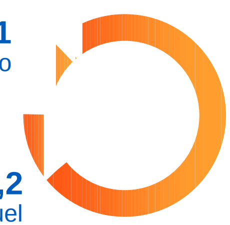
EBITDA 2009

%



Production mix 2009

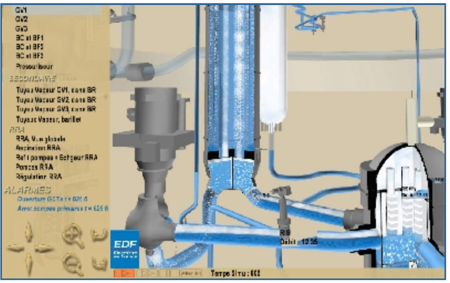
%



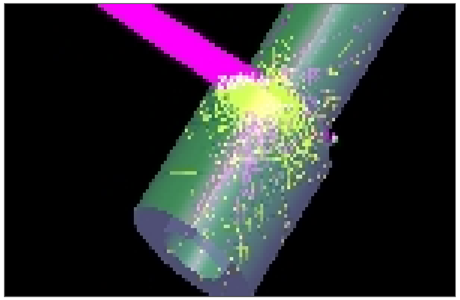
Total : 618,5 TWh

Scientific computing needs at EDF

R&D



Conception



Information technology

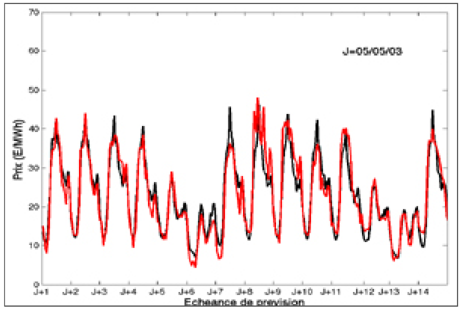
Renewable energies

Electrical networks

Sales

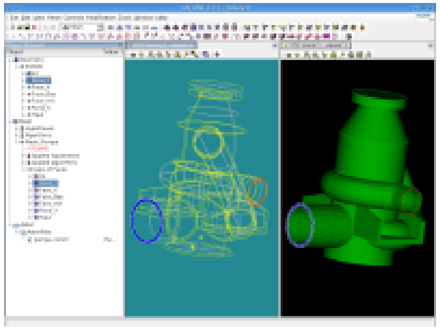
Engineering

Energy management



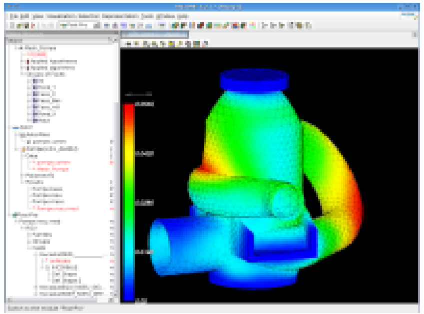
Scientific computing at a glance

- Modeling
  - Approximate reality with a model
  - Often need for a *modeler* to translate a specific case into machine-readable data



- Simulating
  - Execution of a numerical *code* computing the behavior of the model system
  - A whole area of software development
  - Need for the fastest hardware to work on *large arrays of floats*

- Visualizing
  - Results exploration and analysis
  - Need for the best graphics hardware and displays



Improving performance

- **Algorithmics:** reducing complexity before anything else
- **Optimization:** gaining cycles at every line of code
  - Includes work on compilers
- Hardware optimization
  - Reducing latencies and improving bandwidth
  - Make use of Moore’s law

Since this is not enough: **Parallelism**

- Shared-memory: SMP/NUMA architectures
  - POSIX threading, OpenMP
- HPC (Beowulf) clusters
  - One standard protocol: MPI
- **Parallel algorithmics** become the new challenge
  - Get the codes to work with 10000+ CPU cores

Typical user needs

- High-performance computing
  - Massive clusters
  - More or less specialized depending on applications
- Scientific workstation / laptop
  - Modeling and visualizing
  - Results analysis and statistics
  - Accessing the HPC clusters
- Computing chains
  - Servers or small clusters
  - Regular execution of the same code
  - Coupling with other components

One single solution: **CALIBRE**.

A brief history of Calibre

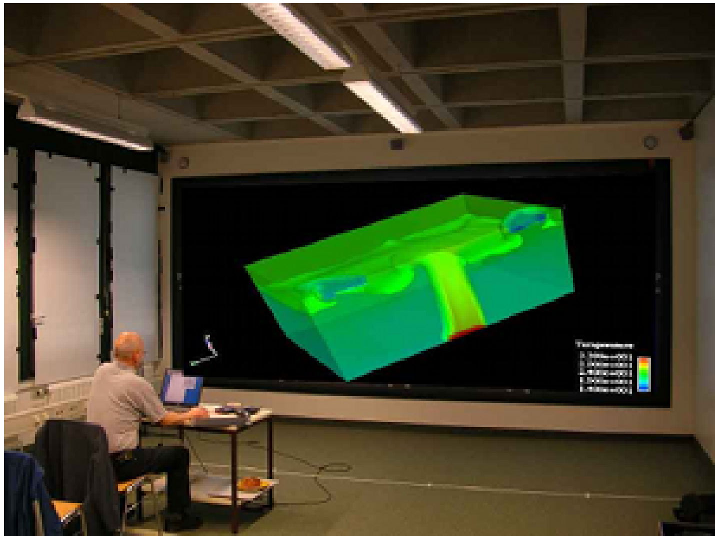
- 2000–2002: Red Hat
  - First solution that worked
  - Same OS for clusters and workstations
  - Automated installation: KickStart
  - Growing concerns with Red Hat stability and maintenance in 2003
- 2003: migration to Debian
  - Independence from editors
  - Reduced kernel / userland adherence
  - Installation: FAI
  - Still a R&D project
- Starting from 2005: industrialisation
  - Official group-wide solution for scientific computing
  - Formal development cycle
- Transfer to IT
  - Integrated offers with SLAs, support delays, etc.
  - Referenced hardware
- Improved development schedule
  - Full-fledged Debian derivative
  - Calibre 6: 1 year after the lenny release
  - Calibre 7 target: 1 month after squeeze

Scientific computing perimeter in 2010

- 3 versions in use
  - Calibre 5 – etch
  - Calibre 6 – lenny
  - Calibre 7 β – squeeze
- 1000 high-end workstations
  - 2/3 R&D
  - 1/3 Engineering
  - ε – the rest

- Clusters
  - Clamart 2: 272 nodes, 25 Tflops
  - Several others at R&D
  - Engineering: 4 clusters, largest 11 Tflops
  - Optimization/trading: several small clusters
  - Blue Gene P cluster (110 Tflops)
  - Red Hat cluster (200 Tflops)

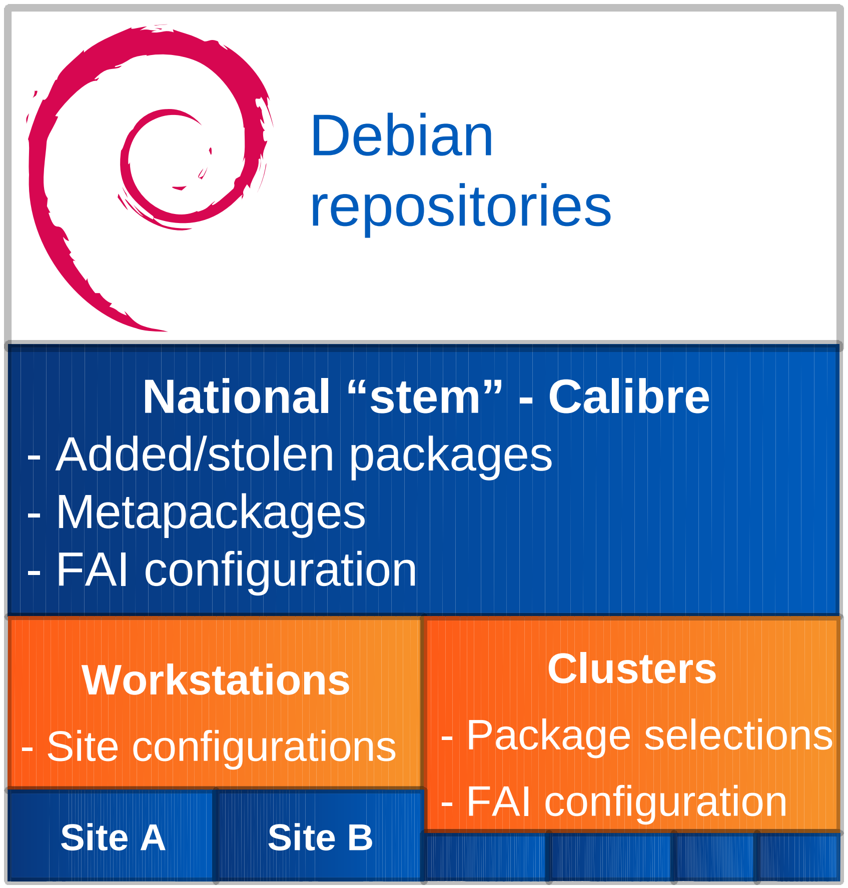
- Graphical cluster
  - 6×3m image wall, 24 Mpixels



Why we keep Debian

- One OS for desktop and clusters
  - Full binary compatibility
- Appropriate release cycle
  - 2 years is fine, 3 years would be better
  - Sole competitors: SLED & Ubuntu LTS
- Largest scientific software offering
  - Only Ubuntu matches, by following Debian repositories
- Designed for customization
  - Custom repositories, easy deployment
- Community openness
  - Easy to get interesting changes into the distribution
- Easy to integrate applications
  - Cool packaging helpers
  - Abundant documentation

In-house development organization



- One repository to rule them all
- Each level has its own support team
- Added packages:
  - Metapackages: only way to maintain consistency across upgrades
  - Backports / additional software
  - Configuration packages: violate policy hard
- FAI classes:
  - Hardware
  - Site
  - Basic package selections
- Site-specific infrastructure:
  - Authentication
  - File servers
  - Print servers
  - …

Our contributions

EDF codes go open source

- No industrial secrets in most of the codes themselves
- Easier collaboration process
  - No contracts, NDAs, copyright assignments…
- Enlarge user community
  - National and international recognition
  - Larger feedback and better validation
  - Lead the development of new standards for the industry

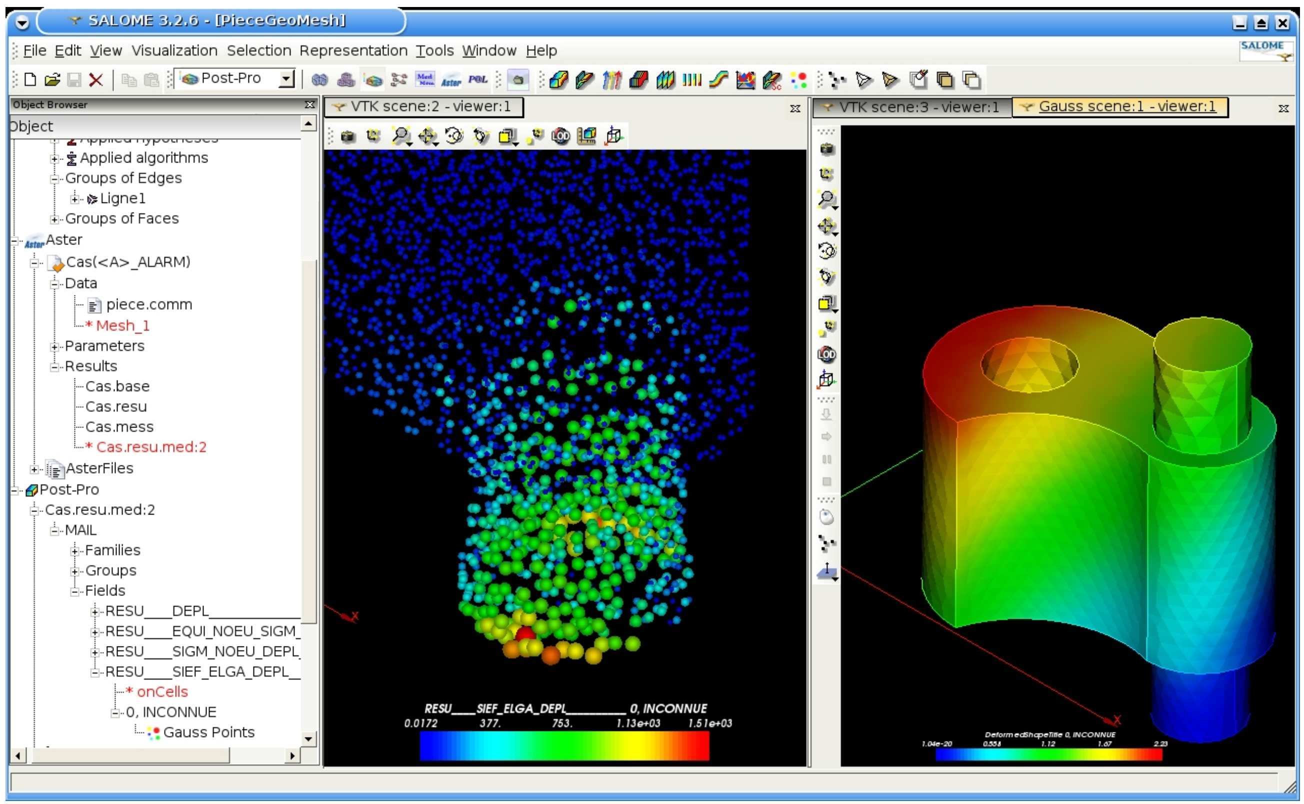
- Rather new: distribution development
  - One of the largest FAI users
  - Regular FAI contributions
  - Feedback and minor contributions for desktop environments
  - Interest in long-term security support: enough for a stronger commitment

Open source codes: Salome

SALOME

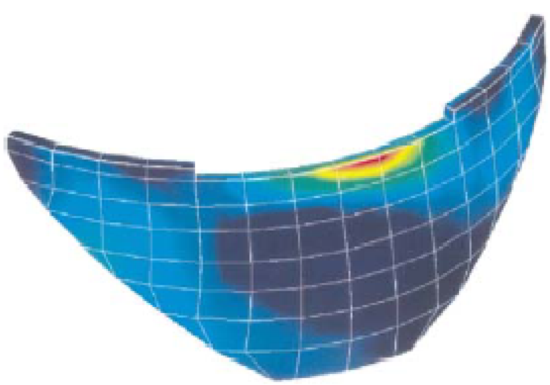
The Open Source Integration Platform for Numerical Simulation

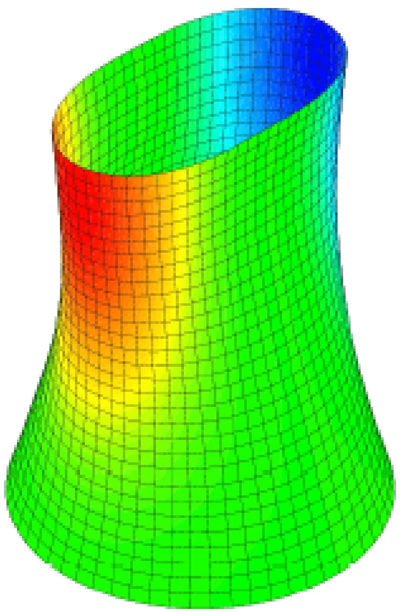
Integrates codes in a single platform:
- modelling
- coupling
- visualization

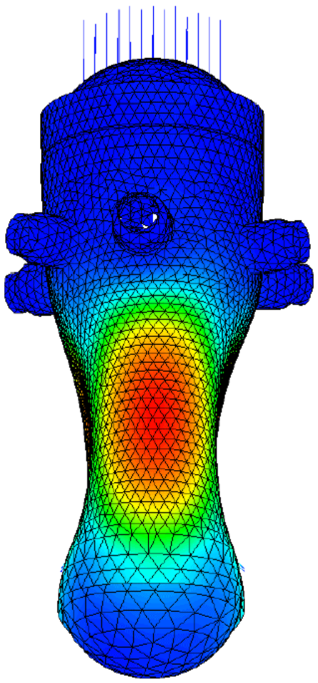


Open source codes: Code_Aster

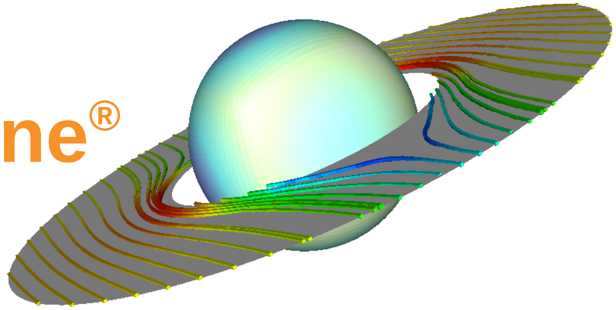
- Originally, a solution to comply with specific requirements of nuclear industry
- Became full-featured thermo-mechanics software
  - Structure ageing
  - Seismic analysis
  - Thermo-hydro-mechanics
  - Acoustics, metallurgy…
- No less than 1 Mloc



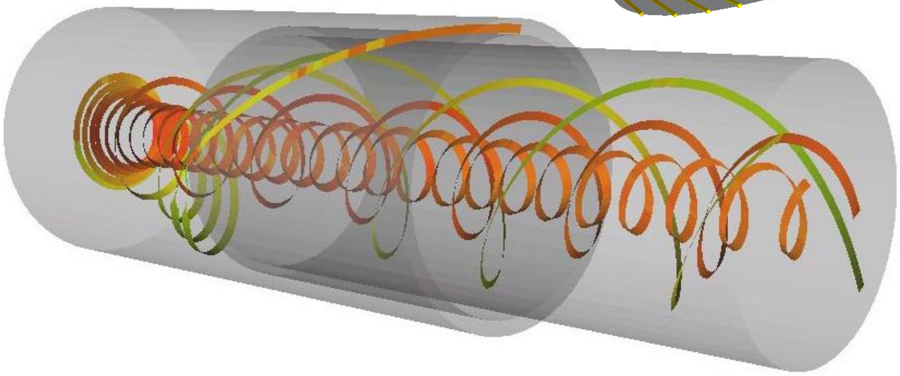




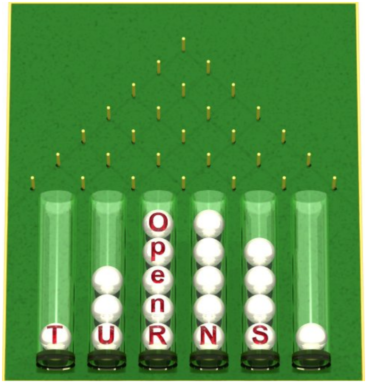
Open source codes: Code_Saturne®



Thermo-hydraulics code:
- turbulence
- combustion
- flow



OpenTURNS



Treatment of Uncertainties, Risks & Statistics
- Add uncertainties to existing codes
- Deterministic computations → probabilistic
- Applications from mechanics to finance

Partnership with EADS & Phimeca

EDF

Example results

- Optimization of security margins for compliance with certification authorities
- Faster and cheaper development of new dams
- Reduce downtime on existing reactors
- Prove that a lake ecosystem can be restored without stopping the hydro plant
- Save millions by planning consumption and production weeks in advance
- Elaborate cheaper photoelectric materials
- Validate network architectures for intelligent electrical meter
- …

Challenges

Some feedback from using Debian in a large corporation

Distribution life cycle and hardware support

- Lifespans:
  - A workstation: 3 years
  - An engineering project: 10 years
  - A nuclear reactor: **30-50 years**
- We need long release cycles
- Long-term security support
  - Currently done in-house: time-consuming and imperfect
  - Very hard to impossible for some desktop components
- Hardware qualification
  - Same hardware for 6-12 months
  - Manufacturers change specifications
  - Issues shared with Windows world
- Call for bids mechanism issues
  - "Of course it works on Linux"
- Kernel obsolescence
  - Operating modern hardware with etch can be hard
  - High hopes for 2.6.32 long-term support
- Graphics drivers
  - So far, only nVidia
  - KMS could change that
  - Again, what about LTS?

Not alone in the world

- Buying software off the shelf
  - Generally for Red Hat
  - Usually works regardless
- Interacting with partners
  - Different reference OSes
- Buying a HPC cluster
  - Every manufacturer has its own solutions
  - No binary compatibility between clusters
  - We integrate Debian clusters in-house
- Integrating a graphical cluster
  - Several proprietary solutions
  - OS compatibility is very sparse

Improving the development environment

- Hard to focus on supporting a single solution
  - Both KDE & GNOME
  - Several IDEs to support
  - 100 users = 50 text editors

- Put the user in control
  - Experiment with aptd and software-center

- Everyone wants his pet compiler
  - Standardization on GCC
  - Other alternatives for application developers

- Integration of profiling tools
  - Proprietary tools are intrusive, expensive and complex
  - Free tools are buggy and hard to package
  - Need for a specific kernel
  - Tools can only be used with a long enough beard
  - Next possible step: getting more feedback by sharing those packages

Integrating with the information system

The good

- Network infrastructure
- Sharing data
- Printing

The bad

- MS Office documents & macros
  - OOo on Windows experimentation
- Corporate websites for IE6 only

The ugly

- Proprietary VPNs
- Bluecoat proxy
- Lotus Notes
  - Even MS Exchange would be easier
- Adobe Flash

Current and prospective solutions

- VMware player
  - Full Windows installation on each machine
  - Heavy and costly
- Remote Windows access (RDP/ICA)
- Remote Linux access (NX/VNC)
  - No decent 3D support

Change resistance

- From other IT members
  - Only 1% of total workstations
  - A slow but strong tendency towards open standards
- From geek users / developers
  - They don't like managed solutions
  - They like short release cycles
- From old-timers
  - Started with SunOS long ago
- From management?
  - They like managed and clean solutions
  - Specific answers to specific needs

Financial considerations

- TCO of a scientific workstation = 2-3 × TCO of a Windows desktop *(very, very rough estimation)*
- This is a **good figure** given we have:
  - Much more features, better hardware
  - Dedicated hot-line & support
  - Only 1% of total workstations
- Not only is Debian efficient, but it is cheap.

- Support the company's profits through R&D
  - One computing result = scientific computing budget for 10 years

Any questions?

Appendix: scientific computing

Hardware/software adherence

- CPU pipelines
  - Raw performance, meaningful for simplistic codes
- L1, L2, L3 caches – latency and amount
  - The amount of data you can handle at once
- Central RAM – latency, bandwidth (FSB), proximity (NUMA)
  - Kernel-level optimization is fundamental
- Communication subsystem
  - Infiniband: 1-5 µs latency, 10-80 Gb/s bandwidth
  - IP: at best 30 µs latency
  - RDMA: handle everything in userspace
  - Network topology counts
- Storage subsystem
  - Disk: 5 ms latency, 1 Gb/s bandwidth
  - Loads of cache, RAID arrays
  - Parallel / distributed filesystems

The software stack

- The kernel: Linux for 91% of Top 500 (Windows: 1%)
- Infiniband support: standardization with OFED
- MPI: standardization of OpenMPI
  - Still many proprietary implementations, few incompatibilities
- Clusters: resource manager
  - Torque/Maui, Slurm…
- Workstations: 3D drivers
  - Only one serious offering: nVidia
- Graphics cluster software: remote 3D
  - VirtualGL, HP RGS…
- Filesystems
  - NFS & Lustre have the lead