Using Linux Clusters as VoD Servers

Víctor M. Gulías Fernández
gulias@lfcia.org

Computer Science Department
University of A Corunha

funded by:
Outline

• Background: The Borg Cluster

• Video on Demand. Introduction

• Goals and initial design proposal

• Key technologies

• Design refinement

• Conclusions and ongoing work
Antecedents

- 1994. **Functional paradigm** + Parallel Computation
  - Declarative, no side-effects
  - High level of expressiveness, abstraction and prototyping

- 1995. Yale Haskell concurrent extension
  - Explicit concurrency, higher-order communications

- 1996-1999. RTS of a **distributed functional language**
  - Semi-implicit model (annotation, dynamic load balance)
  - Explicit model, higher-order asynchronous communication
  - Interesting architecture: **Beowulf Clusters**
Borg, LFCIA's Beowulf Cluster

1 FORERUNNER 3810 (External world)
2 10 Mb Ethernet link
3 Dual Pentium II 350Mhz 384MB RAM 8GB HD SCSI
4 AMD K6 300Mhz 96MB RAM 4GB HD IDE (23, up to 47)
5 100 Mb Fast Ethernet link (2 per node)
6 3COM SuperStack II 3300 Switch (4, 24 slots per switch)
7 1 Gb link (2 independent networks)
Video on Demand. Requirements

• A Video On Demand (VoD) server is a system that provides video services in which a user can request any Video Object (VO) at any time.

• Applications: Movie on demand, distance learning, interactive news, etc.

• Main requirements:
  – Large storage capacity
  – High bandwidth
  – Predictable response time (in preference: low)
  – Support a high number of concurrent users
  – Scalability
  – Adaptability
  – Fault tolerancy
  – Low cost
Existent (Commercial) VoD Solutions

- Project start point 1999 (MPEG-2 2Mbps [LAN], Real 28 Kbps [WAN])
  - Apple, Quicktime Streaming Server (now Darwin Streaming Server)
  - IBM, DB2 Digital Library Video Charger
  - Microsoft, NetShow Theater (now Windows Media Server)
  - Oracle, Video Server
  - Real Networks, RealVideo Server
  - SGI, WebForce MediaBase (now Kassenna)
  - Sun, StorEdge Media Central
  - Some solutions as extensions or adaptations of other commercial products

- Expensive, closed, no scalable, no flexible (adaptable) solutions
VoDka: Distributed Functional VoD Server

- Hierarchical storage system, based on Linux Clusters built with commodity hardware.

- Design using GoF’s desing patterns and OTP behaviours.

- Development languages:
  - Monitorization, Control, Scheduling: Erlang/OTP
  - Input/Output: C (prototype in Erlang/OTP)

- Emphasis in reusability:
  - Use of OpenSource tools (Linux, Erlang/OTP, ...)
  - Adaptation of existent solutions (Darwin)
  - Subsystems independendization (OpenMonet, Erlatron)
• Three level hierarchical structure:
  – Tertiary level (*Storage*): Large capacity
  – Secondary level (*Cache*): Scheduling, aggregated bandwidth
  – Primary level (*Streaming*): Protocol adaptation
Borg Configuration as VoD Server

**Borg0**
- 3COM Superstack II 3300
- 3 SCSI: 2 heads, 1 arm
- 15 slots (33GB Tapes)
  - ~100 2 hour long 300Kb/s movies
  - ~15 2 hour long 2Mb/s movies
- Each head: 5MB/s

**AutoPAK VXA Tape Charger (~$8,000)**
- 3 SCSI: 2 heads, 1 arm
- 15 slots (33GB Tapes)
  - ~100 2 hour long 300Kb/s movies
  - ~15 2 hour long 2Mb/s movies
- Each head: 5MB/s
Key Technologies: Erlang/OTP

- Designed, developed and used by Ericsson AB for their complex distributed control systems
- Suitable for distributed, fault tolerant, soft real time systems
- Main design features:
  - **Functional**: no side-effects
  - **Concurrent**: Asynchronous message passing, value transparent transport, high order communications
  - **Distributed**: Transparent location of processes over nodes
  - Fault tolerance built-in primitives
  - Code hot swap **without stopping the system**
- **Open Telecom Platform**: libraries design patterns for distributed systems
  - Generic servers
  - Supervision mechanisms
  - Distributed database (location transparency, fragmentation, replication, integration with the language)
  - Integration: SNMP, ASN.1, Interface with C, Corba, Java, HTTP
Key Technologies: Linux Cluster

- **Group experience** with high speed networks, distributed systems and clustering over Linux

- **Source code** availability: modification of any part of the software for adapting, correct, locate problems or instrumentation.

- **Homogeneous licence (GPL)**: easy legal treatment (versus, e.g., current MPEG4 situation, where each component has a different licence, and its interaction is sometimes annoying and even contradictory).

- **Compatibility**: code developed with Linux can be ported without problems to Solaris, AIX, Tru64, IRIX, etc. Respect of standards (POSIX 1003.*, SVID, 4.xBSD)

- **Good performance**

- Wide availability of **development tools**

- Different **hardware platforms support** (x86, Alpha, SPARC, ARM, S/390 (zSeries), IA64, SH3, MIPS...)
Learned lessons

- **Need for flexibility**: hierarchical architecture redefinition, giving another module composition with a standard API.
  - The number for levels and the way they are composed should be flexible (e.g.: the whole server as data source of another server)
  - In each level: different protocols both for storage and transference

- **Need of factorization**: communication process abstraction (*pipes, transfers*), server reflection (*VoDka Browser*), monitorization (*observer+mediator*), etc.

- **Independence** with regard to protocols and file formats (at this time in constant change)

- Bottlenecks are not so in the network but in the nodes: **disk access**

- **TCP/IP** is convenient for inter-module communication, but is a **heavy protocol** when dealing with hundreds of Mbit

- **There is no efficient off-line massive storage** with commodity hardware. Tape chargers are efficient but too expensive (IBM 3575: $15000).
Design refinement. Current proposal (I)

Separation of the management subsystem (common web application) and the video server kernel.

The figure represents SMIRNOFF (Second Monitoring and Improved Release Now Offering Further Features) version structure.
Design refinement. Current proposal (II)

Internal transferences in VoDka after a HTTP request from a remote client

A whole server can be data source for the storage level, using the adequate input at the storage driver
The figure shows VoDka SMIRNOFF’s flexibility: 1 streaming level, n cache levels, and the massive storage level.
Design refinement. Current proposal (IV)
XSL transformations using the Cluster

- XML documents (generated with OpenMonet) transformation
  - XSL selection based in the document type, browser, etc.: mod_xsl
  - XSLT C++ Library Sablotron adaptation: sablotron_adapter
  - Load balancing: erlatron

---

```
ab -c 10 -n 500
ab -c 15 -n 500
ab -c 25 -n 500
ab -c 50 -n 500
```

---

```
ab -c C -n 250
```
Conclusions and ongoing work

- Need of flexibility
- Need of factorization
- Independence with regard to protocols and formats
- Bottleneck: disk access
- TCP/IP: heavy protocol
- Doesn’t exist commodity massive off-line storage

- Prototype implantation within the University Campus
- Adaptation and implementation study of a prototype within R network
Fundings

• Related projects:
  – FEDER 1FD97-1759 (VoD server development) (Technological partner: R)
  – XUNTA PGIDT99COM10502 (performance evaluation)
  – University of A Coruña 2000-5050252026 (design patterns in distributed functional systems)

• Infrastructure:
Web resources


A Coruña, September 12-14, 2001
http://www.lfcia.org/sit01