Communications Research Centre Canada
An Agency of Industry Canada
Centre de recherches sur les communications Canada
Un organisme d'Industrie Canada

- SCA ADVANCED FEATURES - OPTIMIZING BOOT TIME, MEMORY USAGE, AND MIDDLEWARE COMMUNICATIONS

Steve Bernier, M.Sc.,
Senior Architect & Project Leader,
Advanced Radio Systems,
Communications Research Centre Canada

Outline

1. Introduction
2. Boot time optimizations
3. Memory footprint optimizations
4. Middleware optimizations
5. Future SCA Core Frameworks
6. Conclusion

1. Introduction

- **The 1st Generation of SCA Core Frameworks**
  - Most CFs come from the JTRS early Steps (2a, 2b, 2c)
  - Implement SCAv2.0, most CFs are not full-featured
  - SCARI-Open
  - Still trying to understand the SCA specification
  - Long boot times, use lots of memory
- **The 2nd Generation of SCA Core Frameworks**
  - Implement SCAv2.2
  - Most CFs have been through JTAP testing
  - SCARI2-Open
  - SCARI++ designed for embedded systems
  - Faster boot times, still use lots of memory

1. Introduction

- **The 3rd Generation of SCA Core Frameworks**
  - Implementation of SCAv2.2 and 2.2.2
  - Much faster boot times, much smaller memory footprints
  - SCARI-GT
  - Include lessons learned from early SCAv2.2 platforms
    - Provides a number of optimizations

- **Optimizations fall into 3 categories**
  1. Boot time
  2. Memory usage
  3. Middleware performance

2. Optimizations for faster boot times

- **For flexible SDR, you need a file system**
  - Applications are made of several components
  - Components are made of several files (XML profiles, Binaries)
  - Running an application implies the deployment of all the application components
  - Bottom line: the SCA is very file-intensive
- **Common boot time issues**
  - Checking for the Integrity of a file system takes for ever!
  - Embedded file systems are very slow

2. Optimizations for faster boot times

- **Addressing the problems associated with file system integrity**
  - Make most of your file system READ-ONLY
  - Get a robust file system driver
  - Use a journaling file system
- **Speeding up file access**
  - Make the DomainManager use the native file system when possible

Test Scenario	File Size	Time without acceleration	Time with acceleration	Improvement
Linux Desktop 3Ghz Pentium without NFS	4 MB	355 ms	20 ms	~94%
INTEGRITY PPC405 SBC using NFS	1.5 MB	2.5 sec	1.5 sec	~40%

2. Optimizations for faster boot times

- **Speeding up file access (continued)**
  - Allow DeviceManager to perform an optimized registration
    - DeviceManager provides digested information to DomainManager
    - Can save over 19 CORBA calls per registering Device, including calls to read remotely from slow file systems

Test Scenario	Standard Registration	One call Registration	Improvement
Linux Desktop, 1 Device	0.56 sec	0.19 sec	~ 66%
Linux Desktop, 4 Devices	1.53 sec	0.24 sec	~ 84%
LynxOS PPC405, 1 Device	0.86 sec	0.13 sec	~ 85%
LynxOS PPC405, 4 Devices	2.33 sec	0.22 sec	~ 91%

2. Optimizations for faster boot times

- **Speeding up file access (continued)**
  - Allow ExecutableDevice to use native file access
  - Allow ExecutableDevice to use a file system cache

Test Scenario	File Size	Cache miss W/O local file acceleration	Cache miss with local file acceleration	Cache hit
Linux Desktop 3Ghz Pentium without NFS	4 MB	355 ms	20 ms	9 ms
INTEGRITY PPC405 SBC using NFS	1.5 MB	2.5 sec	1.5 sec	35 ms

3. Optimizations for faster boot times

- **Optimizing the parsing of XML profiles**
  - A Core Framework must read several XML profiles during the boot up and deployment of an application
  - One option is to use Xerces-C COTS parser
    - Slow and big
  - Another option is to use digested XML profiles
    - Small and fast
    - Relies on proprietary format
    - Digested format can be generated by tools or on-the-fly by a Core Framework

3. Optimizations for faster boot times

- **Optimizing the parsing of XML profiles (cont'd)**
  - Yet another option is to use a hand-crafted XML parser
    - Small and fast
    - Does not rely on a digested format
  - Following table provides metrics for a test to read a .prf.xml file containing 50 properties

Parsers	Static Memory	Dynamic Memory	Parsing Speed
Xerces-C++	3,000 KB	66 KB	6.7 ms
Digested profile reader	300 KB	8 KB	1.1 ms
Specialized profile reader	420 KB	10 KB	1.7 ms

3. Optimizations for faster boot times

- **Smaller footprints: address-space collocation**
  - SCA components are made of SCA interface implementation, CORBA stubs and skeletons, and OS system calls
  - Same kind of SCA components are largely made of the same pieces
  - Up to 70% of the pieces are the same for all SCA Resources
    - Using a ResourceFactory provides enormous footprint savings
  - Collocating the DomainManager and the DeviceManager into a single address space saves 50% of the footprint
  - A DeviceFactory is a must for SCA NEXT!

4. Middleware optimizations

- **The need for middleware**
  - Every distributed system needs a form of middleware to allow communications between different software entities
  - The middleware for SCA is CORBA
  - CORBA supports pluggable transports that shields the developer from the actual transport APIs

4. Middleware optimizations

- **Speeding up communications between components**
  - The following table presents metrics gathered running a ping test using different pluggable transports
    - Does not require changing a single line of source code to switch transport

Parsers	Double Sequence		Octet Sequence	
Number of Elements in the sequence	1024	2048	1024	2048
Average Round Trip Time in usec for PPC405/INTEGRITY using TCP/IP	3334	7272	1428	1767
Average Round Trip Time in usec for PPC405/INTEGRITY using INTCONN	2215	4728	1042	1273
Average Round Trip Time in usec for PPC405/INTEGRITY using direct method invocation thanks to a ResourceFactory	244	492	155	231

5. Future SCA Core Frameworks

- **The next generation of Core Frameworks will provide static deployment optimizations**
  - Instead of optimizing each task required for the deployment of an application, what if we could skip some tasks?
  - A Core Framework could keep information regarding previous decisions taken to deploy an application and skip several steps
    - File caching is a form of static deployment optimization

5. Future SCA Core Frameworks

- **The next generation of Core Frameworks will provide static deployment optimizations (cont'd)**
  - Once an application has been mapped to a platform, why not use a static ApplicationFactory?
    - Can easily be generated by tools
  - Static deployment optimization can provide more deterministic behavior that can be validated
    - It can also help reduce time to deploy an application and the memory footprint requirements
  - SCARI-RT Core Framework

6. Conclusion

- With the latest Core Frameworks, an SCA Radio can now boot in a few seconds
- The requirements in memory represent a fraction of the first generation SCA platforms
- With the assistance of specialized tools, it's only going to get better…

Questions?

steve.bernier@crc.gc.ca

www.crc.ca/sdr

The Communications Research Centre of Canada (CRC)