Big Software for SmallSats: Adapting cFS to CubeSat Missions

Alan Cudmore, Gary Crum, Salman Sheikh, James Marshall

NASA Goddard Space Flight Center

[alan.p.cudmore, gary.a.crum, salman.i.sheikh, james.marshall-1]@nasa.gov
Outline

• Motivation
• What is cFS?
• Experience: CSP / CeREs
• Experience: Dellingr
• Performance
• Future Work
• References

• NOTE: All images courtesy of NASA
Motivation

• Expanding requirements
  – Science
  – Risk tolerance
  – This stresses software (and teams!)

• Budgets are not expanding

• “Small” Satellite does not mean “small” software

• Solution: a trusted framework with reusable components
cFS: core Flight Software

- NASA recognized a need to move away from “Clone and Own”
- Developed to tackle the very issues that SmallSats now face
- Framework and core services (cFE)
- Common set of applications and libraries
- (McComas, 2012) (Fesq, Dvorak, 2012)

“The cFS follows a product line approach with the goal to support systematic reuse.”

(Ganesan, Lindvall, Ackermann, McComas, Bartholomew, 2009)

“At Goddard the main driver for changing the development process is cost, [...] An obvious way to reduce cost and schedule is to increase the amount of software reuse.”

(Wilmot, 2006)
Framework and Core Services (cFE)

- Layered architecture
- Supports Publish / Subscribe Applications
- Events
- Tables
- Time

<table>
<thead>
<tr>
<th>cFS Applications / Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
</tr>
<tr>
<td>HS</td>
</tr>
<tr>
<td>MM</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Applications / Libraries</th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cFE Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exec</td>
</tr>
</tbody>
</table>

Operating System Abstraction Layer (OSAL)

Operating System (Linux, RTEMS, VxWorks, FreeRTOS)
Libraries and Applications

- Currently 12 Applications are available (http://cfs.gsfc.nasa.gov/)
- Optional, depends on mission needs.
- Easy to create
  - Sample application demonstrates messaging, events, and application loop
Heritage

- cFE:
  - Lunar Reconnaissance Orbiter
  - Living With a Star / Radiation Belt Storm Probes

- cFS
  - Global Precipitation Measurement
  - Magnetospheric MultiScale
  - Lunar Atmosphere and Dust Environment Explorer
CHREC Space Processor

- Space Test Program, Houston 5 / ISS SpaceCube Experiment Mini
- CHREC Space Processor Experiment
- NSF Center for High-Performance Reconfigurable Computing
- Presented here last year (Rudolph et al, 2014)
- Two CSPv1 in tandem
  - Xilinx Zynq 7020
  - Arm Dual Core Cortex A9 and Artix-7 FPGA
- Runs cFS!
- Launch 2016
cFS on the CHREC Space Processor

- Work spread over 3 employees
- Created 11 custom applications / libraries
- Code is in well defined applications
- Vary in level of reusability
- This is in addition to existing cFS functionality
CeREs

- Compact Radiation Belt Explorer
- MERiT: Miniaturized Electron and pRoton Telescope
- Flight computer is a CSP
- cFS used for flight software
- (Kanekal, 2014)
Dellingr

- **Hardware:**
  - ARM7 processor (40 Mhz 2Mb RAM)
  - Reaction Wheels
  - Magnetorquers
  - Sensors (FSS)

- **Science**
  - INMS
  - Magnetometer
  - Thermal Louvre
Dellingr and cFS

- Work spread over three employees
- Ported OSAL to FreeRTOS
- Integrate with GomSpace software
- Custom
  - Hardware Library
  - Hardware telemetry
  - Radio
  - ACS
  - Science instruments
- Generated using David A. Wheeler's 'SLOCCount'
Performance

- cFS imposes some performance costs
- Compared build with just FreeRTOS vs cFS
- Code available: https://github.com/jcmarsh/cpek

<table>
<thead>
<tr>
<th></th>
<th>FreeRTOS</th>
<th>CFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dhrystone (per second)</td>
<td>11300.7</td>
<td>10576.4</td>
</tr>
<tr>
<td>WhetstoneDhrystone (KWIPs)</td>
<td>865.7</td>
<td>852.1</td>
</tr>
<tr>
<td>Hardware ping (per second)</td>
<td>757</td>
<td>621</td>
</tr>
</tbody>
</table>
Application Communication Costs

- cFS supports publish / subscribe message passing through the software bus.
- Adds functionality to FreeRTOS queues, increases overhead.
- Chart shows round trip messages passed between two applications.

FreeRTOS vs cFS Message Passing

<table>
<thead>
<tr>
<th>Message Size (bytes)</th>
<th>Free RTOS</th>
<th>cFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>1600</td>
<td>400</td>
</tr>
<tr>
<td>32</td>
<td>1400</td>
<td>400</td>
</tr>
<tr>
<td>64</td>
<td>1200</td>
<td>400</td>
</tr>
<tr>
<td>128</td>
<td>1000</td>
<td>400</td>
</tr>
<tr>
<td>256</td>
<td>800</td>
<td>400</td>
</tr>
<tr>
<td>512</td>
<td>600</td>
<td>400</td>
</tr>
<tr>
<td>1024</td>
<td>200</td>
<td>400</td>
</tr>
</tbody>
</table>
Future Work

- 42 Simulator integration: http://fortytwospacecraftsimulation.sourceforge.net/
- cFS SDK
- Man Rated
Summary

• cFS is a mature framework
  – Strong heritage
  – Reduces personnel requirements
  – Available on a variety of platforms
  – Well suited to CubeSat missions
• Open Source (http://cfs.gsfc.nasa.gov/)
• Already being used on NASA CubeSats
References


