iRobot®

Robots that make a difference

MOOS on Low Power Consumption CPU'S

Frank Agius, Lead Research Software Engineer

iRobot Maritime Systems

Agenda

- Nekton Research vehicles 2007
  - Ranger RN2
  - Transphibian
  - Fast mover
  - “New” Ranger
- Legacy Code Base circa 2007
- Challenges
- The Way forward
- Porting MOOS
- How did it do?
- CPU loading
- CPU Comparisons
- Porting MOOS (again)
- Conclusions
- Cool stuff

iRobot®

Ranger RN2

- 17 kg, approximately 1.5m long
- 4.875” diameter (A-sized hull)
- 1.5 to 2 knots operating speed
- Development platform
  - Feature-based Navigation
  - Sensor evaluation (sonars)
  - CONOPS evaluation
- Features
  - 900 kHz BlueView sonar
  - 1.4GHz Pentium-M processor
  - Vectored thruster
  - Inflation module for emergency recov
  - 190 Whr batteries
  - WHOI micromodem

iRobot®

Transphibian

- Biomimetic propulsion
- 6 DOF motion
- Approximately 30 kg, 1m long
- 1.5 knots operating speed
- Features
  - Oscillating foil thrusters
  - 900 MHz blueview sonar
  - Gumstix Verdex CPU
  - Flood tanks or variable buoyancy engine
  - Camera
  - Autonomous or operator-in-the-loop
  - WHOI micromodem

Contract No. N00014-05-C-0277
Contractor Name: iRobot
Contractor Address: 8 Crosby Dr., Bedford, MA 01730
Expiration of SBIR Data Rights Period: April 15, 2013

iRobot®

Fast Mover

- Propeller propulsion, 4 fin control
- Operating speed > 6 knots
- 3” diameter hull
- ~ 1 meter length
- PXA 250 CPU

iRobot®

“New” Ranger

- Propeller propulsion, 4 fin control
- Operating speed 3-5 knots
- PXA 270 CPU
- Modular software and hardware
- Configurable Payload

iRobot®

Legacy Platform Code Base Circa 2007

- **Ranger**
  - Centralized DB
  - Monolithic
  - Embedded XP
  - Flexible API to the remote pilot
  - Serial connection, No user interface
- **Transphibian**
  - Monolithic
  - Threaded model
  - Unix sockets for message passing
  - Primitive user interface
- **Fast Mover**
  - Multiple Monolithic modules
  - Threaded model
  - Message queues for message passing
  - No user interface
- **New Ranger**
  - Prototype code only

iRobot®

Challenges

- Duplication of code and effort across platforms
  - Multiple drivers for the same hardware
  - Bugs must be mixed in multiple places by different teams
  - Unique calibration routines for each vehicle
- Learning curve for each vehicle
- Inconsistent external interfaces
- Limited debug tools
- Limited simulation
- Low power, small form factor CPU needed for most platforms
- Not enough time or resources or money

iRobot®

The Way forward – Oct 2007

- Goals
  - Reuse best features of Ranger, Tphib and fast mover
  - Common Base Architecture
  - Shared drivers
  - Shared libraries
  - Open API's
  - Modular design
- Implementation
  - Adapt the basic concepts of the Mission Oriented Operating Suite – MOOS
  - Adapt a common processor subsystem – gumstix Verdex (CPU PXA 270)

iRobot®

Porting MOOS

- Work began Oct 2007
- Buildroot development environment for kernel and file system utilities
  - Linux kernel 2.6.21
  - uClibc C library, optimized for embedded systems
- Crosstool for toolchain build
  - GCC 3.45
- Fedora 6 build system
- MOOS V7.01
- Building base MOOS code
  - One compile failure, stropts.h not included with uClibc
  - Reported problem 10/5/2007, fixed 10/8/2007 in V7.02
  - No other build issues
- Ported Ranger RN2 navigation and helm to preserve existing interfaces

iRobot®

How did it do?

- Common code base across 3 (now 4) hardware platforms
- Simplified learning curve
- Robust solution – MOOSDB has never crashed
- Rich set of tools for debug and visualization
- Single source base for Linux arm, Linux x86 and Windows binaries

But…….

CPU loading becomes an issue…..

iRobot®

CPU Load Average

CPU Load average: The number reported (Load average) is the number of blocking processes in the run queue averaged over a certain time period (1min, 5min and 15min). A blocking process is a process that is waiting for something to continue. Typically, a process is waiting for the CPU, Disk I/O or Network I/O.

CPU percentage: The number reported (%CPU) is the amount of a time interval (that is, the sampling interval) that the application was found to be active on the CPU. If the application %CPU is reported at 45, then 45% of the samples showed that application was active on the CPU. The rest of the time the application was in a wait. It is important to remember that a CPU is a discrete state machine. It really can be at only 100%, executing an instruction, or at 0%, waiting for something to do. There is no such thing as using 45% of a CPU.

CPU load average should not be confused with CPU percentage. CPU load average is measured because it does not include any processes or threads waiting on I/O, networking, databases or anything else not demanding the CPU. It narrowly focuses on what is actively demanding CPU time. This differs greatly from the CPU percentage. Load averages measure the trend in CPU utilization over time (vs instantaneous snapshot for CPU percentage) and load averages include all demands for the CPU not just how much was active at the time of measurement.

The load averages shows by increasing duration whether the physical CPU is over- or under-utilized. The point of perfect utilization, meaning that the CPU is always busy and, yet, no process ever waits for the cpu, is the average equal to 1.0 for the 1 minute load average.

CPU load average can be obtained by the user space command uptime:

```
> uptime
 08:46:58 up 1 day, 3:25,  1 user,  load average: 0.11, 0.21, 0.34
```

iRobot

CPU Loading

- CPU load average > 1.0 when running simple MOOS on PXA 270
- Helm and logging found to be the highest consumer of CPU resources
  - Extensive use of floating point math the likely cause
  - PXA 270 uses software emulation to implement floating point math
- Mitigation
  - Reduce logging (not desirable in a debug environment)
  - Move processes to payload processor (if available)
- Long term -> faster CPU
  - Dec 2008 Gumstix announces Overo processor board, based on the TI OMAP 35xx CPU

iRobot®

Gumstix Board Comparisons

	Gumstix Verdex	Gumstix Overo
CPU	PXA270	OMAP 3503
Core	ArmV5	Arm Cortex A8
Vendor	Marvel	TI
Speed	400-600Mhz	600Mhz
RAM	64Mbytes	128Mbytes
Flash	16Mbytes NOR	256Mbytes Nand
Power	~ 1 watt	~ 1 watt
Dimensions	20mm x 80mm	17mm x 58mm
Float support	Software	Hardware
Float Benchmark	7 Mega-flops/sec	19 Mega-flops/sec

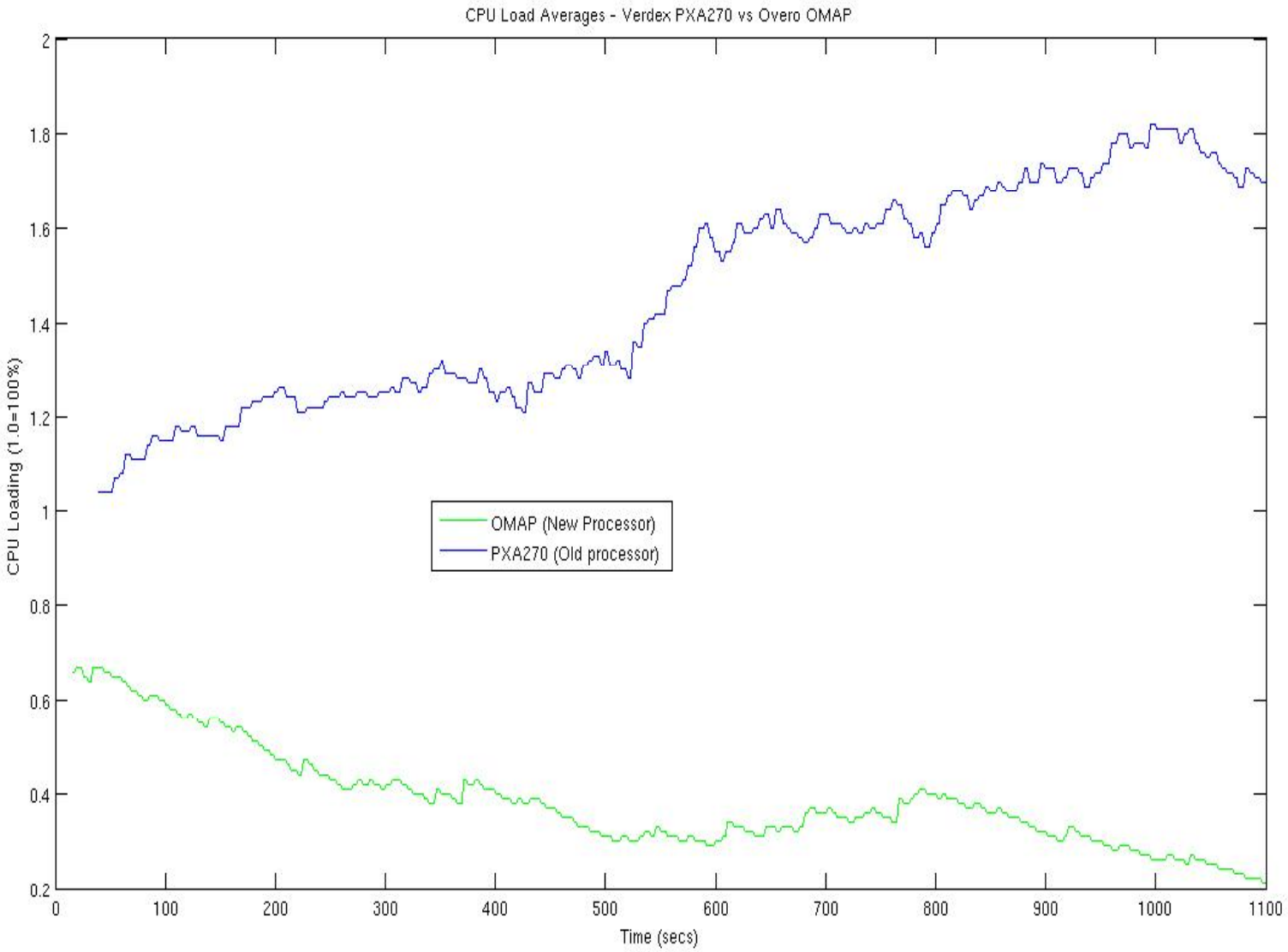
iRobot®

Porting MOOS (again)

- Work began Jan 2009
- OE development environment
  - Linux kernel 2.6.28 (now 2.6.34)
  - glibc C library
  - gcc 4.2 compiler built by OE
- Ubuntu 9.04 build system
- Building base MOOS code
  - No issues

iRobot®

CPU Load Average Comparison running MOOS



iRobot®

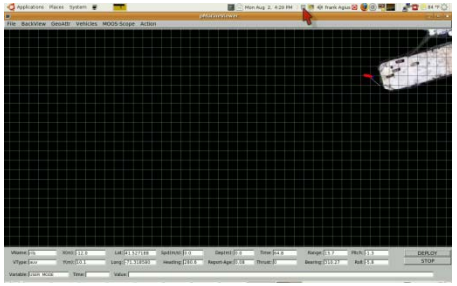
Conclusions

- Omap processor is source code compatible with high level vehicle application code (MOOS and Maritime systems developed code)
- Floating point is 2-3 times faster on the Omap processor
- A/D and PWM hardware on Omap simplify hardware design
- CPU loading while running MOOS well below 1.0 threshold (plenty of headroom)
- Overo board with the Omap processor adapted on Transphibian platform the summer of 2009
- All iRobot research platforms now running on the Overo

iRobot®

Cool Stuff

- Hybrid surface/UUV vehicle
  - High speed transit: > 6 knots
  - Hybrid surface craft/detachable UUV
  - Vectored thruster
  - Autonomous or joystick



iRobot®