lightweight bootchart

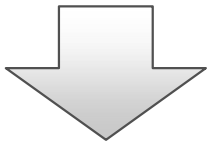
Aug. 31, 2007

Fujitsu Software Technologies Limited
Shuuji Miyake

Motivation

FUJITSU

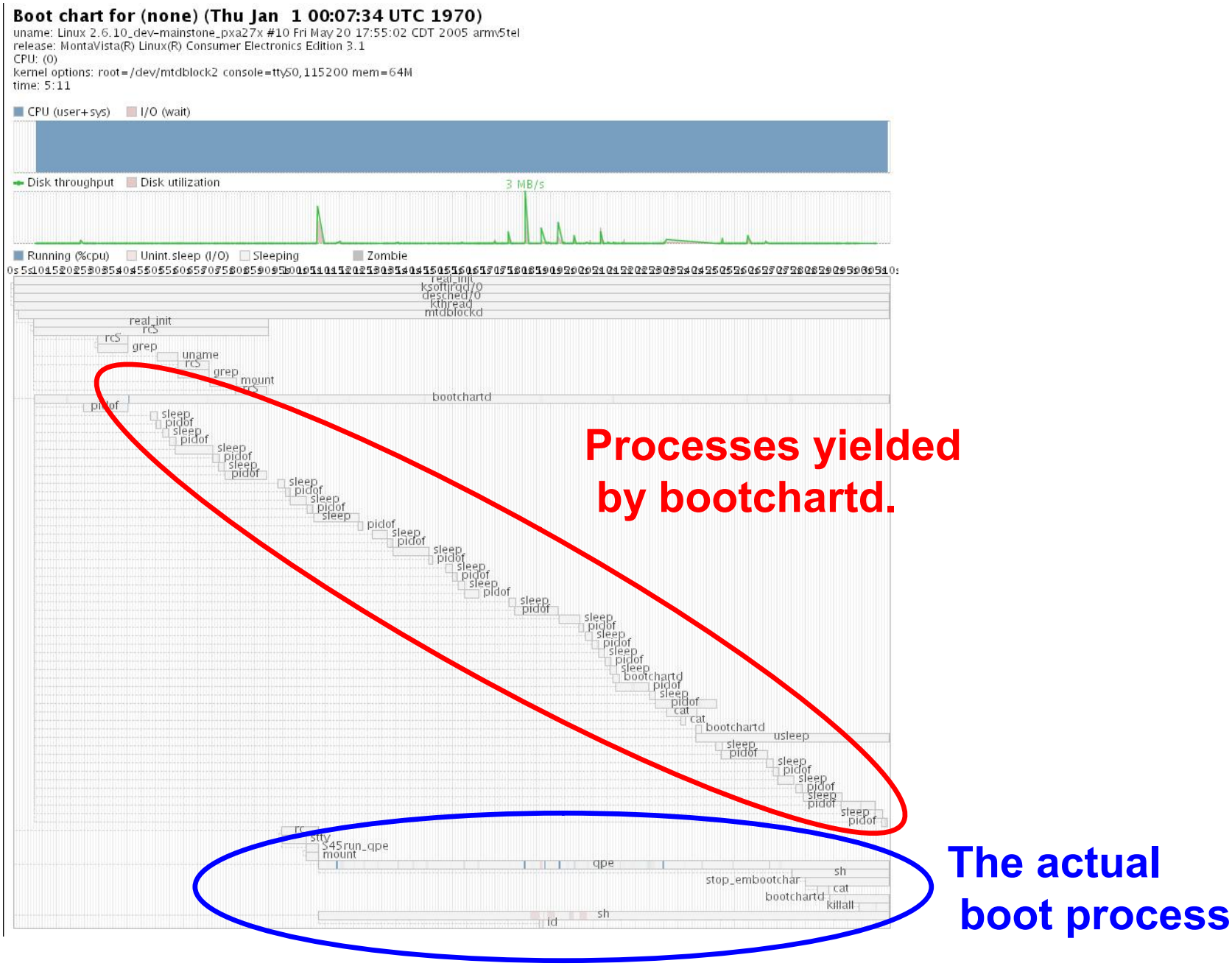
- bootchart (http://www.bootchart.org)
  - is a tool for visualization of the GNU/Linux boot process.
  - corrects process information, etc with shell scripts (bootchartd).
  - **Issues**
    - The overhead of data correction makes difference from what the boot process should be.
    - especially in the case of poor CPU.



Needs of lightweight version!!

Case study (PXA27X)

FUJITSU

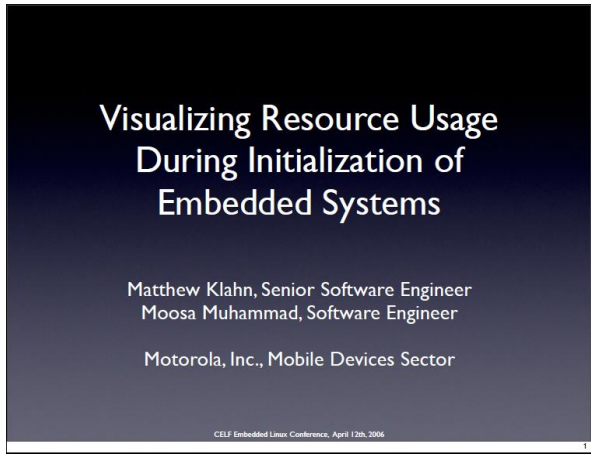


From “Visualizing Resource Usage During Initialization of Embedded Systems” by Matthew Klahn (Motorola, Inc.)

Similar activity

FUJITSU

- embootchart
  - lightweight version of bootchart for embedded space.
  - written in C++.
  - **Status**
    - neither source nor binary availabe.
  - **History**
    - Matthew Klahn from Motorola made a presentation at CELF ELC 2006.
      - http://tree.celinuxforum.org/CelfPubWiki/ELC2006Presentations?action=AttachFile&do=get&target=VisualizingResUsageDuringBoot.pdf
      - “Will publicize on CELF mailing list. This should happen fairly quickly (weeks, not months)”...
    - It became a topic in Celinux-dev ML, but there was no progress.
    - tried to contact him, but got no response...



Let’s build from scratch!!

Why bootchartd sucks

FUJITSU

- written in a shell script.
- A lot of processes are executed and exited.
  - executes cat command periodically to output data to log files.
    - cat /proc/*/stat >> proc_ps.log
  - executes pidof command periodically to determine whether the specific process is created or not.
    - bootchartd will be terminated when mingetty or getty is created.

Strategies (1)

FUJITSU

- implement in C.
  - writing and optimizing string operations in C will achieve better performance because data formats are semi-fixed.
- use SIGUSR1 for stopping bootchartd.
  - reduce the big overhead of watching the creation of specific process.

Strategies (2)

FUJITSU

- reduce system call overhead such as open/close, ...
  - writing data to log files.
    - open once, write many times.
  - reading "/proc/PID/stat" files.
  - use inotify system call to detect the creation and deletion of PID/ directories for avoiding directory search each time.
    - open /proc/PID/stat at the creation time,
    - lseek and read repeatedly until the process exits.

Strategies (3)

FUJITSU

- store data to files in tmpfs.
  - Reduce overhead of real I/O devices such as HDD and flash.
- keep compatibility of data formats.
  - no need to prepare another parser and renderer.
- no need to modify kernel.
  - The same usability as the bootchart.

What's next?

FUJITSU

- Publication
  - CELF internal? (Mailing Lists? Wiki? Any good SCM?)
  - as a Sourceforge project?
  - incorporating into the bootchart project?
- Need help!
  - co-development, evaluation, etc.

FUJITSU

THE POSSIBILITIES ARE INFINITE