12393 Caminito Vibrante
San Diego, CA, 92131
(858) 356-8195
gjinki@gmail.com

Kimyoung Jin

Do {Design – Document – Implement – Test – Deploy } while (True)

Technology

Linux/Real-Time OS, Kernel, Firmware, Linux and Android DLKM(Dynamically Loadable Kernel Modules) - WIFI host driver, WIFI platform driver, WIFI, Power Saving, ARM, Nand Flash Memory, File System, DBMS, MySQL, SQL Server, SQLite, JTAG, GDB, WinDBG, ASW, Google Cloud, TCP/IP, 802.11, HTTP, Flask, Rest API, Bootloader, UART, GPIO, BSP

Tools: git/github, Jenkins, Jira

LANGUAGES

C/C++, Python, Shell Scripts, MFC, C#, Java

and

Can pick up any languages

EXPERIENCE

Qualcomm, San, Diego, CA – Senior Staff Software Engineer

Jan 2012 – PRESENT

Designed and developed automation tools with a primary focus on functionality, scalability, and user convenience. Notable projects include

- Runtime Log Analysis Service for Salesforce
- Crash Triage System
- Linux Ramdump Parser tool for DKMS (Dynamic Kernel Modile) - WIFI host driver
- WIFI/Bluetooth Subsystem Dump parsing tool with TRACE32

Worked with OEMs for Qualcomm WIFI solutions and specifically addressing aspects of focused on Qualcomm BSP/WIFI Platform functionalities – Bring up/Power/Clock and Systems.

- Qualcomm WIFI solutions

Samsung Electronics – South Korea, Software Engineer

July 2007 – Dec 2011

Developed SSD Firmware from beginning of SSD business, focusing on reliability extension and performance functionality.

- Nand Flash Controller Control Layer Software
- Nand Flash Reliability Extension Software

Collaborated closely with Nand Flash Controller RTL design team and Nand Flash Memory team at Samsung Electronics to maximize a performance with multiple channels and minimize a response time and verified FPGA and developed SW simulator for Nand Flash Layers.

- Developed FPGA Software for Nand Flash Controller / SATA interface.
- Nand Flash Controller Control Software Simulator on Windows.

INERVIT – South Korea, System Software Engineer
Mar 2004 – DEC, 2006

Developed Main Memory Based DBMS (Database management System) on Unix/Linux Systems

- Developed SQL Engine

Porting DBMS for Mobile platforms. Worked with Samsung Mobile and SK telecom, EU Carriers – Orange, Vodafone

- Develop APIs for File System, Memory Management for RT OSs.
- Porting to other RTOS (nucleus, psos, REX)
- Samsung Handset Platform (Mocha, Ocean)
- Developed Phones/Organization Applications for EU carriers.

Contractor – South Korea, Software Engineer
2007

- File System Simulator for Mac System
  HFS(Hierarchical File System) to FAT32 Translator on Windows with Samsung Semi conduct
- TRACE32 Automation Controller for MMC devices with Samsung Semi conduct

EDUCATION

Soongsil University, South Korea
Jun. 2007
BS, Computer Science, GPA 3.76/4.5

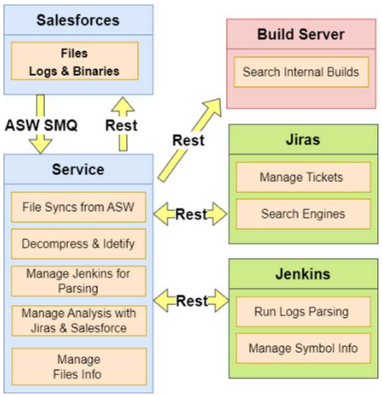
AWARD

Korea National Software Contest of - the Ministry of Information and Communication (2002) - 2nd prize

- Armari : integrated intranet-system in cyber apartment.

PROJECTS

Runtime Log Analysis Service with Salesforce



Implemented a robust runtime log analysis service integrated with Salesforce, streamlining collaboration between Qualcomm and OEMs. In the previous legacy model, the process of issue reporting and log file management was inefficient, involving manual file downloads and elf information matching.

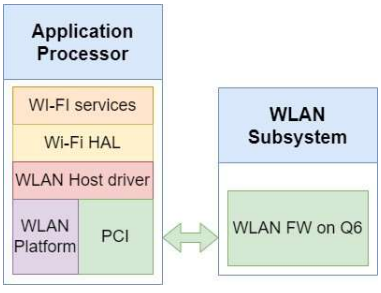
Transformed the workflow by deploying an innovative log parsing system. The system leveraged AWS (Amazon Web Services) for file synchronization, using AWS SQS (Simple Queue Service) to detect and sync new log files to internal servers. Logs were categorized based on type, and parsing tools were initiated using Jenkins automation.

The service culminated in the automatic updating of Salesforce, notifying stakeholders if uploaded symbol files (e.g., Elfs) did not align with their respective logs. Furthermore, the service facilitated the sharing of candidate patches for issue resolution.

Project challenge:

- Concurrency Control: Addressed the formidable challenge of managing concurrent processing of bulk files in multiple cases. Employed Jenkins to ensure effective concurrency control, optimizing efficiency and reliability.
- Load Balancing: Successfully resolved load balancing issues, ensuring smooth and consistent performance throughout the project by utilizing Jenkins, an integral component in maintaining system stability and performance.

Qualcomm WIFI solutions

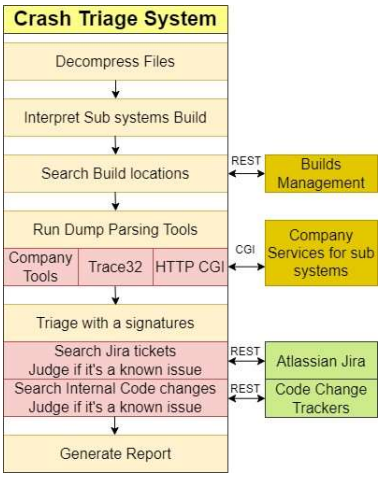


Developed and implemented WLAN Platform driver what controls WLAN FW and controls clock/voltage for WLAN HW for OEM's HW schematic and developed ath10k for Chromebook project.

Debugged and resolved multiple issues from WIFI-Service to WLAN FW and resolved multiple HW timing related issues with ETM and waveform.

Project challenge: Difficult to get a snapshot at a error scene and understand behaviors of HW blocks.

Crash Triage System



Designed and developed the front-end tools to triage a crash dump for Qualcomm MSM, it consisted of multiple subsystems such as Application Processors, MODEM, WLAN, Bluetooth, Audio, RPM(Resource Power Manger).

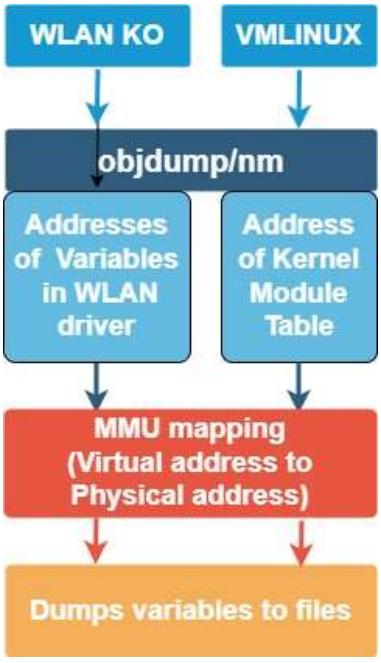
Crafted a versatile solution that automated the process of reading symbol information from dump files, dynamically searching build locations, and parsing relevant subsystem logs (e.g., ADSP, PCI, RPM) to provide a comprehensive overview for debugging purposes.

Facilitated a holistic approach to debugging by interfacing with the company's log parsing servers through CGI/REST protocols, fostering a big-picture perspective for issue resolution.

This front tool seamlessly interacted with JIRA, creating a ticket for issue tracking and searched the error signatures with similar known tickets in jira and code change tracker server. It generated a report with candidate fixes by searching similar tickets.

Project challenge: A accuracy of candidate fixes – build a judgement rule is challenged.

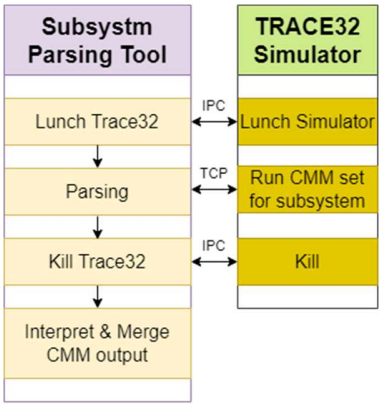
Linux Ramdump Parser tool for DKMS(Dynamic Kernel Modile) - WIFI host driver



Designed and developed a parser tool for Linux Ramdumps, specifically tailored for Dynamic Kernel Module – WLAN host driver.

When Phone user hit a crash, the phone captured DDR snapshot & data immediately. Some data were saved as plain texts in DDR snapshot, potentially exposing private information. Google pixel team requested a parsing tool extract essential WLAN host driver information, facilitating efficient debugging processes while preserving user data privacy. Legacy LRDP didn't support DKMS before Google requested to QC and I had to expand functionalities within LRDP.

Project challenge: Understanding LRDP python scripts and Understanding the mapping logic for Dynamic Kernel Module.



WIFI/Bluetooth Subsystem Dump parsing tool with TRACE32

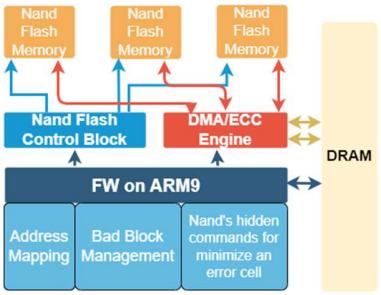
Conceived and designed a front-end tool to streamline the parsing of WIFI and Bluetooth subsystem(firmware/driver) dumps using TRACE32, a debugger and emulator tool. The tool's functionalities included:

- Launching TRACE32(Debugger) remotely.
- Loading dump files and the relevant ELF (Executable and Linkable Format) files into TRACE32.
- Executing PRACTICE Scripts (CMM - Component Monitoring and Manipulation) tailored for WIFI or Bluetooth sub system : These scripts extracted critical variables from the ramdump and stored them in files for subsequent interpretation.
- Organizing activities within modules in chronological order, making the output more understandable and coherent.

The tool significantly contributed to enhancing the comprehension of the control message flow across multiple modules within WLAN and Bluetooth subsystems. By providing a structured view of these subsystems, it facilitated debugging and troubleshooting efforts.

Project challenge: Maintaining CMM Scripts for Multiple Software Product Lines (PLs): A notable challenge involved the maintenance of CMM scripts for different Software Product Lines (PLs). These PLs often had variations in variables and configurations. Adapting and updating the scripts to accommodate these differences across PLs was crucial for ensuring the tool's compatibility with various software

SSD(Solid State Drive) firmware



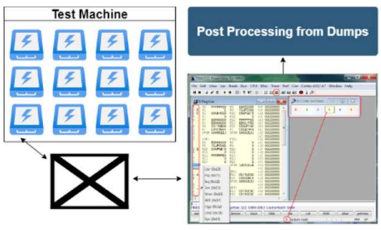
Designed and developed firmware for SSDs on the ARM9 architecture, with a focus on implementing fundamental operating system functionalities over SATA interface.

NAND flash memory had physical limitations - characterized by a finite number of PROGRAM/ERASE cycles, which varies between 10K cycles for MLC (Multi-Level Cell) and 1K for QLC (Quad-Level Cell). Ensured uniform exercise of all physical blocks to extend the lifespan and reliability of NAND flash memory.

Collaborated closely with the NAND Flash memory design team, gaining deep insights - there were multiple hidden ways to get correct data out from wore out cells like adjusting read/erase/program voltage levels on NAND Flash memory and developed these skills on FW to ensure the reliability of data retrieval.

Project challenge: Concurrent Execution with HW Blocks
Confronted the challenge of optimizing firmware to overlap its execution time with the activities of hardware (HW) blocks. This required precise coordination and synchronization to ensure that firmware processes operated efficiently alongside busy HW blocks, ultimately enhancing SSD performance. My team calculated each cycle and execution times for DRAM/SRAM accessing on FW with RTL team and optimize FW as much as possible.

Debugging scripts on TRACE32 for SSD



SSD had no sperate storage to save DRAM and HW blocks snapshot at a failure since the storage size is very big so it required a run time debugging over JTAG.
Test Team run hundreds of devices for each test cycle, Debugging with JTAG took long times by checking global variables on SW.
To save a debugging time and lower a debugging difficulty, I developed debugging script set what run on JTAG debugger – TRACE32. It dumped critical SW data/HW snapshot to files on Host. Script detected an abnormal state of HW blocks which used verified on FPGA and SW bugs which debugged before.

Project challenge: To control HW blocks in TRACE32 script, I had to understand all receipt from HW team. It required extra efforts to understand HW activities. It might be impossible unless I had verified FPGA.