David A. Butterfield

David.Butterfield@acm.org

Software Engineering and Analysis

Fourteen-year veteran of Sun Microsystems

Co-founder, ten-year veteran of Locus Computing Corporation

Experienced — Proficient — Articulate — Intelligent — Analytical — Creative — Passionate

Summary

I have over thirty years of experience creating and supporting advanced operating system software, in roles as architect, designer, code writer, troubleshooter, mentor, technical leader, manager, and consultant. My background includes successful solo projects as well as projects where I was a team member or team leader. I enjoy understanding systems, whether at an architectural level or in the deep details of a complex control algorithm, whether analyzing an existing system or creating a new one. My most personally rewarding activities are when I am working to understand a difficult technical problem or devise an elegant or insightful solution, either myself or in collaboration with others.

Primary High-Skill Areas

- System software architecture and design
- Software engineering and programming
- Analysis of systems and their behavior
- Analysis of requirements / implementation alternatives
- Problem solving, emphasizing root-cause analysis
- Written and oral technical communication

Experience Emphasis

- Linux, Solaris, other Unix kernel implementations
- C programming language
- Reliability and Availability
- Event-driven systems
- Distributed filesystems with replication
- Virtual-machine implementations
- SCSI target and transport protocols
- Networking protocols
- I/O frameworks for networking and storage
- Device drivers and protocol translation modules
- Technical leadership of engineering teams
- Geographically-distributed engineering teams

Education, Professional Organizations

- UCLA, MS Computer Science, 1982
- UCLA, BS Mathematics and Computer Science, *Cum Laude*, Departmental Highest Honors, 1979
- Member, Association for Computing Machinery

Outstanding Abilities

- I rapidly understand complex concepts and systems
- I integrate diverse ideas into a cohesive conceptual model
- I understand systems at multiple levels of abstraction and move my thinking easily among them
- I transform data into information and understanding
- I use appropriate mathematical techniques in my analyses
- I develop and communicate a clear architectural vision
- I can explain complex concepts clearly, with appropriate detail
- I can communicate engineering concepts to non-engineers
- I can make decisions without complete information
- I bring a sense of aesthetics to my software design
- I write code that is understandable and maintainable

Professional Practices

- I strive to understand things accurately and precisely
- I seek measurable data to confirm or disprove hypotheses
- I devise methods to observe / measure systems under study
- I seek root causes
- I share knowledge, ideas, values, and wisdom
- I keep the broader and longer-term context in mind
- I balance long-term interests with short-term needs
- I seek ways to implement and deliver in useful increments
- I bring alternatives, with analysis, to decision-makers
- I advocate for what is right under the circumstances
- I help other people understand things better
- I encourage creativity and experimentation
- I thrive on interesting and challenging technical problems

Publications and Patents

- *iSCSI-SCST Storage Server Usermode Adaptation*, February 2017
  https://davidbutterfield.github.io/SCST-Usermode-Adaptation/docs/SCST_Usermode.html
- *Method and Apparatus for Using a Directory Service to Facilitate Centralized Device Naming*
  U.S. Patent 7,925,872 granted April 12, 2011
- *Bus Specific Device Enumeration System and Method*, U.S. Patent 7,203,774 granted April 10, 2007
- *Efficient Direct Memory Access Transfer of Data and Check Information to and from a Data Storage Device*
  U.S. Patent 6,687,767 granted February 3, 2004
- *Dual Operating System Computer*, U.S. Patent 4,747,040 granted May 24, 1988
- *Display Context Switching Arrangement*, U.S. Patent 4,744,048 granted May 10, 1988
- *A Proposal for 80386 VM86 Performance Enhancements*, presented to Intel Corporation, April 1987
- *A Merged UNIX/DOS Environment on the Intel 80386*, presented at the January 1987 *Usenix* conference
- *Unraveling OS-Merge*, Computer Design magazine, April 1, 1986
- *The LOCUS Distributed System Architecture*, G. Popek, ed., MIT Press, 1985
- *Network Tasking in the LOCUS Distributed UNIX System*, presented at the June 1984 *Usenix* conference

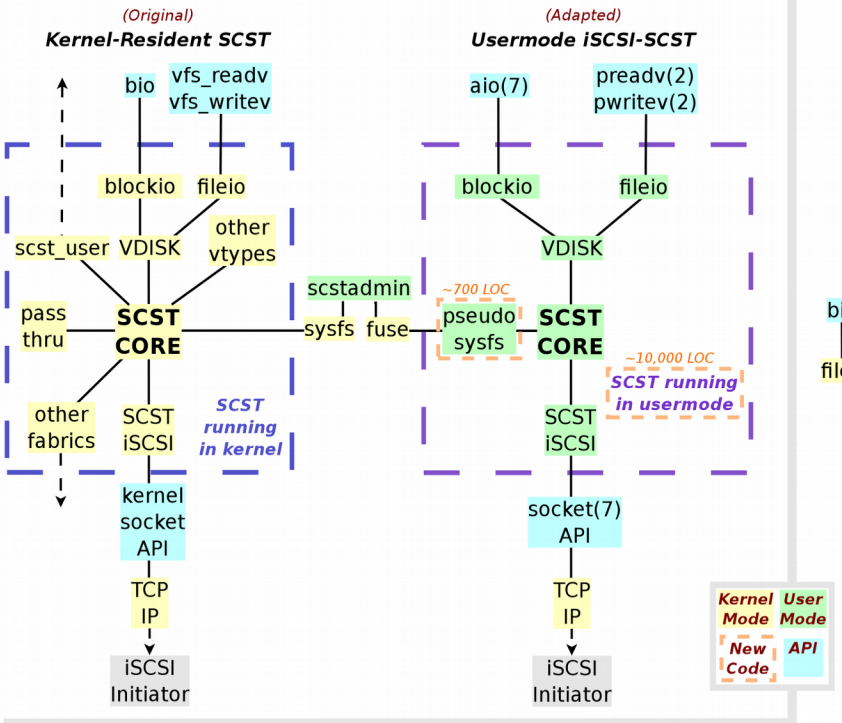
David A. Butterfield
Software Engineering and Analysis

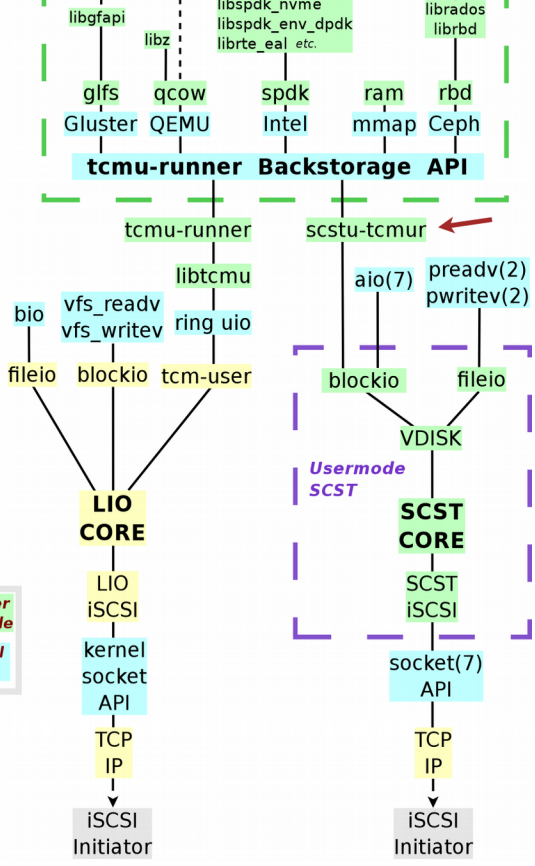
June 2015 – August 2017: Multi-threaded Engine and iSCSI-SCST Usermode Adaptation

(This work was entirely my own)

- Designed and implemented a **multi-threaded event-driven engine** (MTE) to run on Linux or other POSIX systems
  - › MTE emphasizes **high-performance** memory allocation, dispatching, network I/O, and inter-thread queuing
  - › Event loop supports timers and work scheduling using epoll_wait(2) with sockets, AIO events, and signalfd
  - › Source code is entirely in C and can be found at https://github.com/DavidButterfield/MTE
- Adapted the ***SCST*** open-source Linux kernel **iSCSI storage server** software to run entirely in usermode under MTE
  - › Implemented emulation of needed Linux kernel functions in usermode, including simulated software interrupts
  - › SCST server runs the same way it does in the kernel, but on usermode threads in a multi-threaded process
  - › Adaptation to usermode was implemented with virtually no change to 80 KLOC of mature SCST kernel C source
  - › C code is at https://github.com/DavidButterfield/SCST-Usermode-Adaptation
  - › The diagram on the left below compares SCST as configured running in the kernel and in usermode
- Studied and experimented with the **performance** of the usermode iSCSI server on 1 Gb Ethernet
- Developed an informal **mathematical model** to help analyze and explain client-server network performance
- Developed an ***Adaptive Nagle*** algorithm to improve IOPS performance of CPU-bound Read workloads by over 50%
- Wrote a paper (with many diagrams and plots) describing the above work and analyzing its performance
  - › https://davidbutterfield.github.io/SCST-Usermode-Adaptation/docs/SCST_Usermode.html
- Implemented interface module scstu_tcmur which allows SCST in usermode to use ***tcmu-runner*** storage handlers
  - › Wrote a backstorage handler to support disk I/O through **Intel's *Storage Performance Development Kit (SPDK)***
  - › This enables SCST in usermode to utilize backing storage managed by **Ceph, Gluster, QEMU,** or **Intel SPDK**
  - › The diagram on the right shows how usermode SCST can share common backstorage handlers with tcmu-runner

BELOW: The adaptation of the SCST iSCSI server to run in usermode reuses ~80 KLOC of a very mature and efficient implementation of the SCSI SPC/SBC command set and iSCSI protocol, with virtually no change to the existing source code.





RIGHT: The scstu-tcmur module drives SCST storage I/O through the tcmu-runner backstore API, enabling a common set of backstore handlers to be used for in-kernel (LIO) and usermode (SCST) iSCSI servers.

February 2008 – May 2015: Lefthand Networks / Hewlett Packard

(I started at Lefthand Networks, which was acquired by Hewlett Packard late in 2008)

- Was a member of the software development team for the SAN/iQ (StoreVirtual) highly-available storage cluster
- Was responsible for the cluster *"infrastructure",* including facilities for threading/locking, scheduling, socket I/O, memory allocation and debugging, trace/logging, cross-thread communication, network protocols, *etc.*
- Main focus areas: ***Product Performance, Software Observability, Software Quality, Developer Education***
- Each of these four areas required both **detailed technical analysis** and **significant ongoing implementation**

Product Performance and Memory Footprint

- Analyzed performance and developed changes for SAN/iQ 9.5 resulting in a 27% increase in SSD 8KRR IOPS
- Designed and implemented a multi-threading retrofit to the single-threaded cluster software for SAN/iQ 10.0
  - 16-month project, conceived, designed, prototyped, implemented, and delivered by me as a solo project
  - Split of the single thread into two resulted in a second major IOPS increase of 23% (2011)
- Above changes totaled to a 57% IOPS increase, enabling an SSD product with market-viable performance
- Made further optimizations and split off a third thread, seeing another 65% improvement in IOPS (2014)
- Designed and implemented changes reducing RAM required per node by over 2GB, with no reduction in speed
- Devised optimizations reducing program startup initialization time by about half (on the order of minutes)
- Analyzed root causes of *(e.g.)* slow sequential read at medium queue-depths, and developed fixes
- Implemented many other optimizations of various sorts throughout the cluster code (mostly to increase IOPS)

Software Instrumentation and Observability

- Rewrote the stacktrace facility to provide much more and better information, improving the ability to diagnose problems from a single failure (of importance in cases when it is unknown how to reproduce it)
- Added statistics gathering and reporting to the memory allocator to better understand memory usage patterns
- Added sophisticated checking and debugging capability to the memory allocator, including detection of some types of corruption, stacktraces of allocating and freeing callers, detection of use-after-free and some cases of uninitialized use, *etc;* this caught *many* bugs for multiple developers over the years
- Added statistics gathering to the censemble scheduler to measure which parts of the system consume CPU resources under various workloads – on its first day this found a function wasting 8% of a CPU
- Wrote a *"Task Monitor"* tool to facilitate dynamic observation of the major components of a SAN/iQ application running under the censemble scheduler (similar to the *top* program, but with knowledge of program internals)
- Added comprehensive checking, logging, and debugging capability to the socket I/O facility

Software Quality and Maintainability

- Embarked on a *(never-ending but largely successful)* campaign to eradicate memory leaks from cluster software
- Wrote a socket simulation module to allow testing of the iSCSI server in the SAN/iQ synthetic test environment
- Created a maintainable API for non-cluster applications to subscribe to cluster change events
- Did a massive review of incoming code changes, identifying dozens of bugs (and fixes for most of them)
- Debugged many subtle problems, often being asked to assist other engineers experiencing difficulty
- Carried out a *Change Request* (bug report) classification study, examining distributions of various bug subsets
- Wrote scripts to mine and analyze data from the source code control system and integration build/test run logs

Developer Education *(some materials also suitable for non-developers)*

- Created the first (and only) drawings and documents covering the entire SAN/iQ architecture in detail
- Diagrams and Presentations
  - *Introduction to SAN/iQ Cluster Software Architecture*, 13 slides with narration text (2 hours)
  - *Censemble Central Scheduler*, 10 slides with narration text (one hour)
  - *An Abbreviated Tour Through censemble Scheduling*, 21 slides with presenter notes (1.5 hours)
  - *Overview of the Multithreaded censemble Implementation*, 32 slides with presenter notes (3.5 hours)
  - *SAN/iQ Architecture: Components and Connections* (broad view of the entire product on one sheet)
  - *SAN/iQ Cluster Infrastructure* (common facilities for queuing, scheduling, I/O, memory, system calls, *etc.*)
  - *SAN/iQ Cluster Datapath* (functional blocks and connections, 5 slides, detailed and abridged versions)
  - *SAN/iQ Cluster Networking* (with reference points for failure analysis)
  - *SAN/iQ Cluster Models: The Closure* (a fundamental abstraction explained)
  - *SAN/iQ Multithreaded Datapath – Read from Local Store* (OP flow diagram, showing thread divisions)
  - *Memory Allocator Advanced Debugging Features*
  - Many other explanatory diagrams and charts depicting the results of various experiments and analyses

David A. Butterfield
Software Engineering and Analysis

Experience History
1992 - 2007

October 2006 - March 2007: Parascale, Inc.

- Was a member of an entrepreneurial team designing a WAN distributed filesystem
- Investigated architectural approaches to asynchronous file replication with tunable coherency
- Researched existing capabilities of industry-leading products, and studied the Solaris ZFS implementation
- Prepared architectural diagrams and participated in discussions with potential customers and partners

May 1992 - August 2006: Sun Microsystems, Inc.

October 2001 - August 2006: Senior Staff Engineer, Solaris Storage Software, Broomfield, CO

- Was architect for the Solaris kernel storage I/O group in Sun's Network Storage Division
- Led a team to determine requirements and priorities for Solaris storage software
- Proposed and designed solutions to numerous difficult technical problems
- Reviewed Solaris storage software interfaces and proposed interfaces
- Guided engineers in the department to ensure clean interface designs under a consistent architecture
- Provided advice and assistance to department engineers in solving technical problems
- Negotiated technical plans and interdependencies between my group and the Solaris I/O team
- Sponsored technical proposals to the Architecture Review Committee and facilitated their approval
- Participated in technical due-diligence on a storage company and product Sun subsequently acquired
- Developed architecture for SCSI Object Storage Device (OSD) protocol support under Solaris
- Wrote prototype implementations of Solaris kernel interfaces above and below the OSD target driver

November 2000 - October 2001: Senior Staff Engineer, Solaris RAS Analysis, Broomfield, CO

- Was a member of Sun's Reliability, Availability, and Serviceability (RAS) Computer Analysis Laboratory
- Studied the reliability and availability of Solaris software components
- Proposed and designed a project to examine available defect-related data to determine patterns or trends
- Analyzed a subset of high-priority defects discovered in the field to determine
  - when and why each defect came into existence
  - how each defect escaped detection until after the product was in the field
  - what engineering process changes could reduce the number of defects distributed to the field
- Reviewed in detail the design of a data integrity project and raised several issues
- Participated in a team to analyze failures related to Solaris cluster availability
- Wrote short *Introduction to Solaris OS Availability*, intended to be within the grasp of any software engineer

February 1999 - November 2000: Senior Staff Engineer, Solaris I/O Engineering, Broomfield, CO

- Was a member of the team responsible for the Solaris Device Driver Interface (DDI) and I/O frameworks
- Wrote analyses and designs for handling DLPI Style-2 drivers, clone drivers, and SCSI drivers under *devfs*
- Designed device enumeration interfaces and wrote prototype SCSI device enumeration code
- Designed and prototyped a solution to significantly reduce boot time on systems with many SCSI buses
- Wrote a new chapter *Drivers for Network Devices* for the Solaris *Writing Device Drivers* book
- Researched and proposed Solaris high availability models
- Gave presentations to partners on driver hardening, fault reporting, fault isolation, and fault injection
- Delivered a new chapter *High Availability Drivers* for the Solaris *Writing Device Drivers* book
- Reviewed numerous engineering documents and device drivers and produced detailed comments

January 1994 - January 1999: Senior Staff Engineer, Solaris Driver Engineering, Dublin, Ireland

- Accepted an international assignment to Sun's newly-opened engineering office in Dublin, Ireland
- Interviewed, hired, trained, and managed engineers for a Solaris x86 device driver team in Dublin
- Provided support to team members as leader, architect, mentor, and troubleshooter
- Integrated the team into the Sun worldwide engineering community
- Redesigned the interface for the Solaris Generic LAN Driver (GLD) and rewrote the code to deliver GLD v2
  - achieved a 15% improvement in the throughput of GLD-based network drivers
- Provided technical assistance and advice to other engineering groups on the Dublin site
- Smoothly transitioned architectural support and leadership of the team prior to the end of my assignment

May 1992 - December 1993: Staff Engineer, Solaris Driver Engineering, Los Angeles, CA

- Established and managed Sun's first Solaris x86 device driver development group
- Designed and produced the first Solaris x86 Driver Development Kit (DDK)
- Designed and implemented the first Solaris Driver Update mechanism
- Wrote extensive technical documentation for writers of Solaris x86 device drivers
- Worked extensively with a training developer to create materials for a class for driver writers
- Assisted other engineers in the driver development team with difficult technical problems

David A. Butterfield

Software Engineering and Analysis

Experience History
1975 –1992

June 1982 - May 1992: Locus Computing Corporation, Inglewood, CA

(I was one of seven founders of Locus Computing Corporation)

February 1989 - May 1992: Scientist, AIX Product Support

- Was one of two co-chief technical leads for support of IBM's AIX/370 and AIX/PS2 clustered Unix system
- Held architectural oversight of a department of over 70 technical personnel
- Produced technical evaluations of proposed product enhancements
- Resolved architectural issues for other engineers in the department
- Established a small team to propose and prototype product enhancements
- Provided guidance and mentoring to less-experienced engineers
- Implemented changes to departmental processes to increase quality and reduce response time
- Met with customer personnel to understand their current and future needs
- Communicated with architects and engineers in other organizations to coordinate plans
- Handled some of the more difficult customer technical problems
- Provided technical guidance to marketing personnel

June 1984 - January 1989: Development Manager, 8086 Virtual Machine Development

- Was architect *and* manager of Locus Computing Corporation's *UNIX/DOS Bridge* product line
- Designed and produced 8086-virtual-machine implementations running on 80286 and 80386 processors, marketed as *"AT&T SimulTask", "Merge/286", "Merge/386", "AIX PS/2 DOS Merge", and "Sun Merge";* distributed by Locus, Microport, *(the old)* SCO, AT&T, IBM, Sun Microsystems, and other OEMs. That work later evolved into two descendant products known as *"NeTraverse Merge"* and *"Win4Lin" [after my time on the project]*
- Led the team starting with 5, and expanding to 28, development and test engineers and technical writers
- Was responsible for product design, development, testing, documentation, and maintenance
- Estimated required time and staffing, and created project schedules
- Designed the external interfaces and internal architecture
- Developed required Unix kernel changes in C and x86 assembly language
- Wrote low-level interrupt and I/O trapping and emulation code
- Analyzed performance characteristics, and developed optimizations and recommendations
- Gave technical presentations to potential customers, and to the January 1987 *Usenix* conference
- Wrote proposal to Intel for 80386 vm86 performance features, which later (mostly) appeared in the Pentium
- Produced technical analyses of competing products

April 1983 - May 1984: Senior Systems Programmer, LOCUS Clustered OS Development

- Was a member of the team implementing the LOCUS clustered Unix operating system (based on 4.1 BSD)
- Implemented kernel support for remote fork and exec operations to other nodes in a LOCUS cluster
- Implemented prototype support for migrating a running process from one node in a LOCUS cluster to another
- Presented a paper about remote tasking in LOCUS to the June 1984 *Usenix* conference
- Wrote Chapter 4 "*Remote Tasking"* for *The LOCUS Distributed System Architecture* (MIT press, 1985)
- Ported the LOCUS kernel to a 68000-based computer (with another engineer)
  - › Wrote kernel assembly language and memory management support for the 68000-based computer
  - › Wrote kernel support for communication with an off-board I/O processor

June 1982 - April 1983: Programmer, Unix System III Port

- Was a member of a small team that ported Unix System III to a new machine
- I ported all user-mode system utilities except the C compiler

September 1979 - June 1982: University of California, Los Angeles

- Was system programmer and administrator for the UCLA Mathematics departmental computer
- Ported Unix v6 and v7 to a DEC PDP-11/23
- Modified Unix v7 kernel to support older 18-bit-address peripherals on a 22-bit-address memory bus
- Wrote device drivers for graphics devices and for a custom external I/O processor
- Provided Unix system setup and informal consulting to a few departments around campus

June 1975 - June 1982: Consulting and Contracting

- Performed various consulting and contract work, mainly for Teledyne Controls and for the U.S. Navy
- Designed, wrote, and documented a small stand-alone operating system for the PDP-11 computer
- Developed a comprehensive diagnostic program to test a custom data-acquisition hardware prototype
- Wrote real-time vector graphics software in PDP-11 assembly language to drive a Vector General display
- Wrote a variety of device handlers and emulators in PDP-11 assembly language
- Implemented a two-dimensional subset of the 1977 SIGGRAPH CORE graphics standard in FORTRAN