# David A. Butterfield

**Software Engineering and Analysis**

[linkedin.com/in/davidbutterfield](linkedin.com/in/davidbutterfield)
[David.Butterfield@acm.org](mailto:David.Butterfield@acm.org)

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

With deep experience designing and implementing various species of advanced system software, I enjoy learning to understand complex systems, whether at an architectural level or down in the subtle details of a distributed algorithm. It is personally rewarding to solve a difficult technical problem, or discover an insight that leads to clearer understanding or an elegant solution, and to help others do likewise.

## High-Skill Areas

- Software design and development, fluent in C, Go
- Deep understanding of complex systems
- Mathematical thinking and technical analysis
- Investigative experimentation and prototyping
- Problem solving, emphasizing root-cause analysis
- Written technical communication and diagrams

## Experience Emphasis

- System software development
- Linux, Solaris, various Unix kernel implementations
- Performance analysis and improvement, scaling
- Observability, instrumentation, measurement, tracing
- Reliability and availability, interoperability
- Quality and maintainability
- Geographically-distributed engineering teams

## Technologies Implemented

- Multithreaded event-driven systems
- Asynchronous algorithms, closures, synchronization
- Networking and storage protocols, I/O frameworks
- Device drivers and protocol translation modules
- Device Driver Interface (Solaris x86 DDI)
- Driver Development Kit (Solaris x86 DDK)
- Virtual-machine monitor (x86), emulators
- Low-level machine control, embedded systems
- Clustered OS remote tasking (remote fork, exec)
- Distributed, replicated filesystems

## Example Tools, Protocols, APIs Used

- gcc, go, as, gdb, strace, valgrind, git, bash, sed, tcpdump
- Ethernet MAC, IP, TCP, iSCSI, SCSI-SPC, SCSI-SBC, fuse
- Ceph/rbd, Intel SPDK/DPDK, ICE, logic/protocol analyzers

## 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 can explain complex concepts clearly, in 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

## Education

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

## Independent Projects under [https://github.com/DavidButterfield](https://github.com/DavidButterfield)

- MTE *(2016) – High-performance multi-threaded event-driven engine (epoll_wait, socket, aio, timer, work-queue, etc.)*
- usermode_compat *(2017) – Compatibility functions for running some Linux kernel code in usermode*
- SCST-Usermode *(2017) – Port of iSCSI-SCST kernel storage server to run in usermode (SCST source unmodified)*
- Technical paper *(2017) – describes the SCST-Usermode implementation and analyzes its performance*
- DRBD-9.0 *(2019) – Port of Linbit's Distributed Replicated Block Device to usermode (DRBD source unmodified)*
- spdk *(2019) – Intel Storage Performance Development Kit support to run usermode versions of SCST and DRBD*
- dbd *(2021) – Mirrored Distributed Block Device prototype written in Go*

## Patents

- *Method for Using a Directory Service to Facilitate Centralized Device Naming,* U.S. Patent 7,925,872 *(2011)*
- *Bus Specific Device Enumeration System and Method*, U.S. Patent 7,203,774 *(2007)*
- *Efficient DMA Transfer of Data and Check Information to/from a Data Storage Device* U.S. Patent 6,687,767 *(2004)*
- *Dual Operating System Computer*, U.S. Patent 4,747,040 *(1988)*
- *Display Context Switching Arrangement*, U.S. Patent 4,744,048 *(1988)*

*Last Updated August 2021*

**2021: Mirrored Distributed Block Device**
- Implemented a simple prototype distributed mirrored block device in about 3200 lines of code in Go
  - › Mirrors of a block device may reside on different computers in a TCP/IP network (uses Go *net/rpc*)
  - › Consistency maintained using two-phase commit and synchronous logging of metadata changes for recovery
  - › Uses *tcmu-runner* backend storage handlers (via *cgo*); block devices are managed using *targetcli(8)*

**2019: Distributed Replicated Block Device (DRBD-9.0) Port to Usermode and SPDK** *[diagram]*
- Ported Linbit's DRBD Linux kernel implementation to run in usermode
  - › Port can run on a POSIX system without the need to load a kernel module for DRBD
  - › The ported Linbit kernel code is unchanged, with its expected kernel environment emulated around it
- Discovered and reported a few DRBD bugs, found by running *valgrind* and *libusan* on the usermode port
- Ported DRBD and SCST to run under Intel's Storage Performance Development Kit (SPDK)
  - › Implemented interface modules between the SPDK *bdev* interface and Linux kernel Block-I/O *(bio)* protocol
  - › Implemented an interface module to allow use of *tcmu-runner* block storage handlers under SPDK

**2018 - 2019: Zoned Block Devices (ZBD) – Western Digital Research**
- Reviewed and provided detailed technical comments on several revisions of draft T10/T13 ZBD proposals
- Wrote test code to validate ZBD operations against real and emulated devices

**2016 - 2017: Multi-Threaded Engine and iSCSI-SCST Usermode Adaptation** *[diagram]*
- Designed and implemented a multi-threaded event-driven engine (MTE) to run on Linux or other POSIX systems
- Ported the *SCST* open-source Linux iSCSI storage server kernel software to run entirely in usermode under MTE
  - › Implemented emulation of required Linux kernel functions in usermode, including simulated software interrupts
  - › Implemented an interface module to allow SCST in usermode to use existing *tcmu-runner* block storage handlers
- Studied and experimented with the performance of the usermode iSCSI server over Ethernet and through loopback
  - › Developed an *Adaptive Nagle* algorithm to improve IOPS performance of CPU-bound READ workloads by 50%
- Wrote a technical paper describing the above work and analyzing its performance
  - › Paper develops a mathematical model to help analyze and explain client-server network performance

**2008 - 2015: SAN/iQ Storage Cluster – Lefthand Networks** *(acquired by Hewlett Packard)*
- Responsible for the infrastructure of the event-driven *SAN/iQ (StoreVirtual)* highly-available storage cluster *(threading, scheduling, cross-thread communication, memory allocation, I/O, timers, system calls, epoll_wait, etc.)*
- ***Product Performance***
  - › Implemented a clean and transparent multi-threading retrofit to the existing single-threaded event-driven engine
  - › Measured and analyzed system behavior to determine performance bottlenecks, adding observability as needed
  - › Developed several independent optimizations, together increasing IOPS to 2.5 times the starting performance
- ***Software Observability***
  - › Rewrote the stacktrace facility to provide much more and better information, improving first-failure diagnosability
  - › Added statistics gathering and reporting to the memory allocator to better understand and optimize memory usage
  - › Added statistics gathering to the event-driven scheduler to measure CPU consumption of application tasks
  - › Wrote a *Task Monitor* tool for dynamic observation (similar to *top(1)*, but with knowledge of program-internal tasks)
- ***Software Quality and Maintainability***
  - › Added comprehensive checking, logging, and debugging capability to socket I/O and memory allocation facilities
  - › Wrote a *socket(7)* simulation module to allow testing of the iSCSI server in the SAN/iQ synthetic test environment
  - › Created a maintainable API for external applications to subscribe to cluster change events, to replace *ad-hoc* logic
  - › Debugged many subtle problems, often being asked to assist other engineers with bugs resistant to diagnosis
  - › Carried out a *Change Request* (bug report) classification study, examining distributions of various bug types
- ***Developer Education – Diagrams and Presentations*** *(Examples from 12+ in total)*
  - › *Introduction to SAN/iQ Cluster Software Architecture* (13 slides with narration text, 2 hours)
  - › *Overview of the Multithreaded censemble Implementation* (32 slides with presenter notes, 3.5 hours)
  - › *SAN/iQ Multithreaded Datapath* (OP flow diagram, showing thread divisions)

**2006 - 2007: WAN distributed filesystem – Parascale** *(entrepreneurial team)*
- 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 technical discussions with potential customers and partners

**David A. Butterfield** *Experience Summary*
**Software Engineering and Analysis** *Through 2006*

## 2001 - 2006: Solaris Network Storage Software – Sun Microsystems

- Was department technical lead reporting to the Director of Solaris Network Storage Software
- Led the team to determine requirements and priorities for Solaris storage software
- Reviewed Solaris storage software proposed interfaces, providing guidance to department engineers
- Provided advice and designed solutions to numerous difficult technical problems

## 1999 - 2000: Solaris I/O Engineering – Sun Microsystems

- 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
- Gave presentations to industry partners on driver hardening, fault reporting, fault isolation, and fault injection

## 1992 - 1998: Solaris x86 Driver Development – Sun Microsystems

- Established and led Sun's Solaris x86 device driver group, reporting to the Director of Solaris x86 Engineering
- Designed and produced the first Solaris x86 Driver Development Kit (DDK), and Driver Update mechanism
- Accepted an international assignment to establish a Solaris x86 device driver group in Dublin, Ireland (1994-1998)
- Interviewed, hired, trained, and managed developers, and provided technical leadership and support to the team
- Reimplemented the Solaris Generic LAN Driver (GLD) API, increasing throughput of all GLD-based drivers by 15%

## 1990 - 1992: AIX Product Support – Locus Computing

- Was chief technical lead for support of IBM's AIX/370 and AIX/PS2 clustered Unix system
- Held architectural oversight of a department of over 70 technical personnel
- Personally handled or assisted with some of the more difficult customer technical problems

## 1984 - 1990: Virtual Machine Development – Locus Computing

- Designed and produced 8086-virtual-machine implementations running DOS under Unix on an Intel 286 or 386 CPU
  - › Distributed by Locus, AT&T, IBM, Sun Microsystems, Microport, (the old) SCO, and other OEMs
  - › Known as *AT&T SimulTask, Merge/286, Merge/386, AIX PS/2 DOS Merge,* and *Sun Merge*
  - › Later evolved into two descendant products *NeTraverse Merge* and *Win4Lin [after my time on the project]*
- Wrote the low-level interrupt, I/O trapping, and emulation code in C and x86 assembly language
- Analyzed performance, developing optimizations and recommendations
- Wrote *Proposal to Intel for 80386 VM86 Performance Enhancements* *[which later appeared in Pentium Appendix H]*
- Gave technical presentations to potential customers, and at the January 1987 *USENIX* conference

## 1983 - 1984: LOCUS (Clustered 4.1BSD Unix) OS Development – Locus Computing

- Implemented kernel support for remote fork and exec operations to other nodes in a LOCUS cluster
- Implemented kernel support for migrating a running process from one node in a LOCUS cluster to another
- Wrote Chapter 4 *Remote Tasking* for *The LOCUS Distributed System Architecture* (MIT press, 1985)
- Wrote a paper about remote tasking in LOCUS for the June 1984 *USENIX* conference

## Before 1983

- 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
- Designed and implemented a small stand-alone operating system for the PDP-11
- Developed a comprehensive diagnostic program to test a custom data-acquisition hardware prototype
- Ported Unix v6 and v7 kernels from a DEC PDP-11/45 to a PDP-11/23; wrote Unix device drivers in C