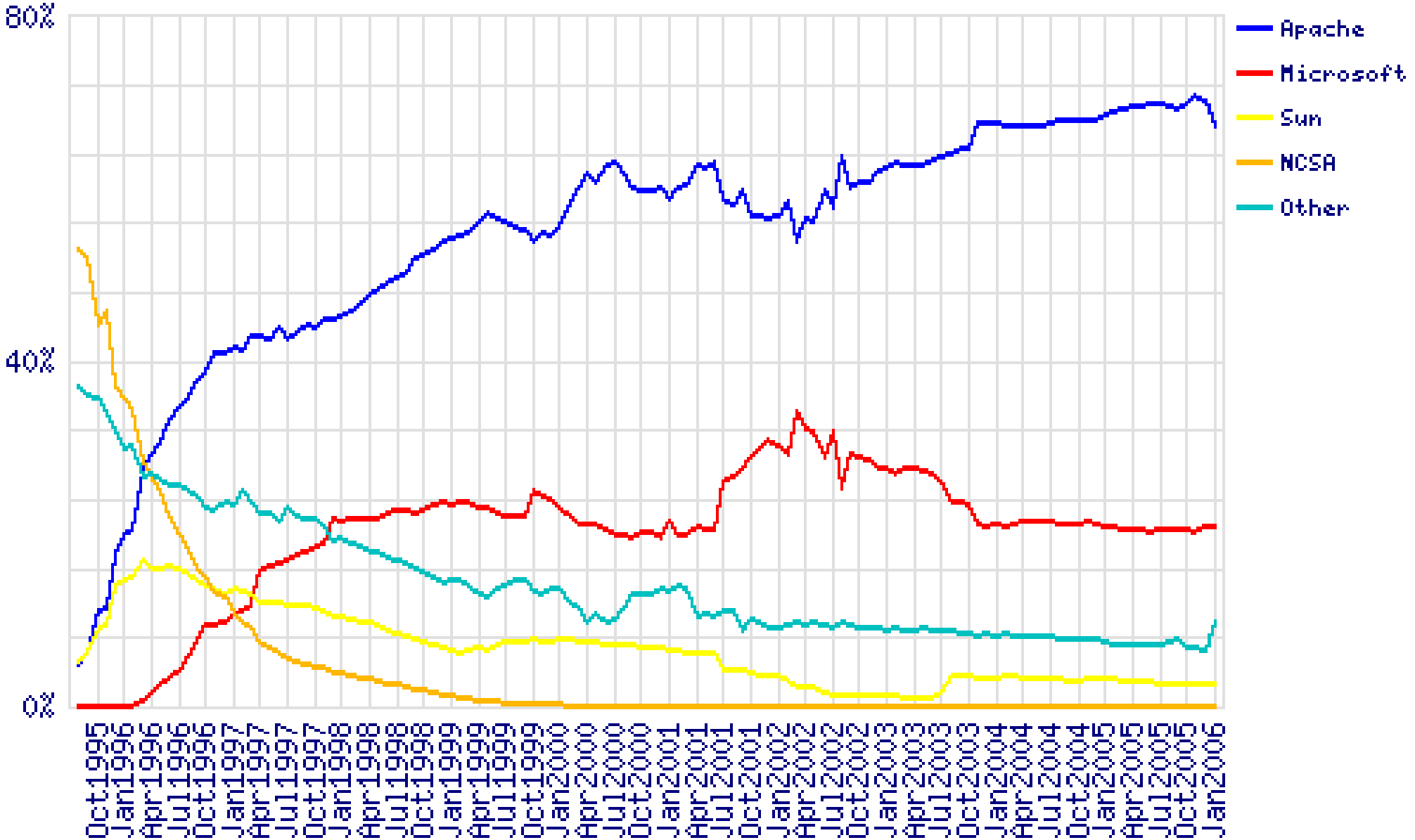
Open Source and the Choice to Cooperate

Brian Behlendorf
Apache Mozilla
CollabNet Subversion

Bootstrapping the Apache Project

- 1995, 8 Webmasters (Wired, IMDB, MIT, early ISPs) had a problem.
- National Center for Supercomputing Applications (NCSA) had released a server, but it was not being maintained.
- We had “patches” to the server for performance, security, functionality.
- We also were all on the HTTP working group of the Internet Engineering Task Force, and wanted to see HTTP become and remain an “open” platform.
- None of us wanted to become full-time web server developers.
- Apache thus started as a “fork” of the NCSA codebase.
- Why did it grow so fast?
- Features, but also flexibility and timing.

Apache's Growth



From Project to Foundation

- By 1998, 60% of the web was running on Apache
- IBM, Sun, and other corporations expressed interest.
- We realized we needed a way to balance corporate interest and involvement with a consensus-driven approach to software development.
- We needed legal protection as well – what if someone decided to sue us as individuals for something found in the code?
- We also needed a way to scale the project to be about more than just a web server.
- Solution: a US-registered, 501c3 non-profit membership-based organization, called the Apache Software Foundation.
- Members elect a Board, the Board appoints executive officers and conducts oversight of the operations of the ASF.
- The ASF holds the right to redistribute work from contributors; including all copyright, patent, and trademark rights.

What does the ASF do?

- The ASF cares primarily about developer communities. Good code will come from effective, well-managed communities.
- 44 top-level projects, 1800 committers
- ASF spends a lot of time educating developers about legal aspects to Open Source software: copyright, patents, contributions, etc.
- ApacheCon, a yearly ASF-produced conference.
- We give companies guidance on how to work with us – making clear we're much more interested in developer involvement than cash donations.
- Legal contributions from volunteer and corporate lawyers.
- However, the ASF maintains a clear separation between “church and state” - companies have no formal role, it's all about the individual developers.
- **No paid staff.** For better and worse.

The Tools We Need and Use

Key requirements:

- Enhance transparency, archiveability
- Work well over the wide-area network

Most important tool: mailing lists

- Discussion is the most fundamental part of writing Open Source
- Web-based discussion forums can also work
- Make sure it is all archived, so decisions can be referred to later.

Second most important tool: a source code repository

- CVS, the “Volkswagen” of version-control tools
- Subversion, a modern replacement for CVS
- Most other code versioning tools just are not suitable.

Issue tracking: Bugzilla, a bit of Jira

Wikis: probably the fastest way to get ideas on a page and organize them, though real docs should be in HTML

Development Methodology at Apache

The ASF has a consensus-based approach

- We make sure the communities are healthy, that there is involvement by more than just one or two people in a project.
- Minor decisions can be made quickly, major decisions require three +1 votes and no vetos.
- Someone issuing a veto has to declare why they are vetoing and what can be changed to address their veto.

Linux and others use the “air traffic controller” approach

- One lead developer acts as a coordinator for the incoming contributions of others.
- Subdivision along functional lines – networking vs. disk drivers vs. memory mgmt. Linus acts as a final check of something aggregated long before.
- He also delegates maintenance of older releases.

Most other projects are somewhere between these two....

Development ~~Processes~~ Principles

- Release schedule is usually time-based.
- Ongoing active development, usually split between an active development branch and a stable bugfix-only branch.
- Unless important bugfixes force a release, generally we look for projects to make a point release once every few months.
- Feature set is “whatever gets done” - much like SCRUM
- Release date is usually preceded by a bug shake-out and a call for testers.
- If a particular feature is not yet mature enough for wider use, it is usually either disabled by default, or kept out of the next release.
- Think of this kind of management as similar to the way an investor might manage a portfolio of investments... or how someone might try to plan a really good party.

Community Management

Imperative #1: Be humble!

- Treat other developers, even (especially) new developers, like peers to the process.
- Have respect for contributors – assume they are intelligent.
- The most scarce resource in an Open Source project is developer motivation, so it must be encouraged and cultivated.

Imperative #2: Be transparent!

- Make decisions as a group, in front of the community.
- Be responsive to challenges to the code, and don't take it personally.

Imperative #3: Think of the user community

- It's too easy in an Open Source community to simply solve your own needs, rather than think about the needs of the broader community.
- Apache strives to be the neutral meeting ground for all developers no matter who their employer is.

Addressing Corporate Involvement

- Most new Apache projects start as efforts of individuals, but as they grow in importance, corporate involvement helps attract a critical mass of developers and activity.
- Our “model” for involvement of corporations is to have them fund specific developers, who then collectively determine the best path forward.
- Those developers can then present their patches or thoughts to their peers in the process, and defend them on technical merits rather than on employer objectives.
- Since our focus is more infrastructural than whiz-bang, a conservative approach works well.
- In essence the developers become “representatives”, but they still are subject to the meritocracy.

Can This Work for Other Communities?

- First question to ask: start a neutral third ground like Apache, or go with one tied to an existing corporation?
- If the developers feel a sense of ownership in the project, they'll be loyal to it, they'll identify with it deeply and care about its future.
- Such passion for the project as a whole is essential to facilitating real collaboration – to put aside egos and agendas and make the right decision for the community.
- Works best if you don't have to make promises to end-users, but can be opportunistic about what is delivered. How? By not trying to predict the future, but instead certify what you've got today.