;login:

THE MAGAZINE OF USENIX & SAGE

February 2001 • volume 26 • number 1

inside:

CONFERENCE REPORTS

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

conference reports

This issue's reports is on the First Workshop on Industrial Experiences with Systems Software (WIESS 2000).

OUR THANKS TO THE SUMMARIZER:

Alan Messer

First Workshop on Industrial Experiences with Systems Software (WIESS 2000)

October 22, 2000

San Diego, California, USA

Summarized by Alan Messer

KEYNOTE ADDRESS

James Gosling, Sun Microsystems

James Gosling presented the keynote address on his experiences with getting the design principles behind creating the Java language out into the real world.

Despite its fairly recent rise to fame, the project which led to Java was begun 10 years ago. Several consumer electronic companies were trying to define software models for their future products to solve portability, interface, and programmability problems. Instead of trying to tackle each of these problems head on, the project looked for a simple, overarching solution.

Despite its aims for a Write Once, Run Anywhere paradigm, today Java is mostly a Learn Once, Work Anywhere language due to the proliferation of different Java versions and profiles adapted to particular environments.

In Java's progression from research project to commercial language, many features got dropped (bad ideas, deadlines, etc.). However, in this process most of the really good ideas managed to stay, along with a few "features." Of those that stayed, the garbage collector is probably one of the nicest features for developers to use on a day-to-day basis, along with features like array subscript checking.

But even today there are many requests for new features to be added to the language. A fairly elegant proposal to incorporate generics (type polymorphism) has been made. There are also proposals to adapt good features from other languages, such as function invariants (Eiffel) and assertions (C, C++). And there are always requests for some of those features that never made it from C, such as enumerated types, not to mention many new APIs for particular domains such as real-time.

Those features which require actual language extensions present a problem, especially if they also require changes to the underlying virtual machine. Thankfully, the virtual machine has managed to stay mostly the same, ensuring compatibility at the interface level, and important features slowly make it, when necessary, into the language itself.

REFEREED PAPERS

SESSION: SYSTEM ARCHITECTURE

Operational Information Systems: An Example from the Airline Industry

Van Oleson, Delta Airlines; Karsten Schwan, Greg Eisenhouer, Beth Plale, and Carlton Pu, Georgia Institute of Technology; and Dick Amin, Delta Airlines

Van Oleson spoke on Delta Airlines' experiences in cooperation with Georgia Tech in adding enhanced information systems to Delta's operational information system.

The project started as a skunkworks project in Delta after a previous project failed to bring the required additional infrastructure to support next-generation airline information services (e.g., gate information).

Existing infrastructure is antiquated at best and large scale (cluster of IBM S/390s), with WAN links (up to ATM) to outlying airports serving 10,000+ flight displays across the country. Adding an order-of-magnitude more displays and enhanced services (connection directions), at low cost, to an operational system presents a challenging task.

This project took the approach of tapping the existing system and deriving event notifications from the existing infrastructure. This enabled them to provide the enhanced services required plus tackle the scalability and availability issues of the existing system.

Using the tapping approach, a secondary piggyback system provides support for the enhanced services and uses modern techniques such as weak multicast and fail-over UNIX clusters to meet the scalability and high availability requirements of the system. To make use of existing network infrastructure in the presence of a large quantity of data, just-in-time XML transcoding was used to compress and translate data from the servers to the flight displays.

This project not only presents many interesting problems similar to those solved by distributed middlewares, but also shows how real-world problems require both integration with existing systems and mind-sets able to meet new requirements in operational systems.

Question: Is Delta collaborating with other airlines on this project?
Answer: This is looked on as a business advantage by Delta, so at this stage there is no interaction with other airlines.

Experiences in Measuring the Reliability of a Cache-Based Storage System

Dan Lambright, EMC

Dan Lambright described work at EMC in measuring the software reliability, maintainability, and availability of a disk cache. What happens when one of the cache lines fails or a software error causes a line to be unreliable?

The problem with such questions is that the limitations in existing tools make failures hard to detect. With this lack of good detection tools, such failures are also typically slow to fix. While this may be less of a problem for small systems, large systems these days have large quantities of cache which can account for up to 32Gb of space. Clearly, with so much caching, availability is a key concern, but how do you measure, detect, and understand it?

This project took the approach of using software fault injection tools to help determine the effect of errors on the system's availability, maintainability, and performance. Errors were injected into the cache maintenance data structures to discover the consequences of those errors and the ability to detect those errors.

This work found that typical existing diagnostic tools were fairly ineffective, since they stopped at the first error detected. Also, existing QA were initially resistant to fault injection techniques. Lastly, they discovered that several errors lead to system unresponsiveness rather than to errors affecting maintainability.

Question: Did you consider data errors too?
Answer: No, this was not considered for this study.

Question: Since programmers always think tests are error free, is coverage limited with programmer-designed tests?
Answer: Yes. Developers are good, but development-group-based tests (testing each other) are better. Also, developers get feedback (pagers) on software/test problems.

HP Scalable Computing Architecture

Arun Kumar and Randy Wright, HP

Arun Kumar presented his experiences in moving computer architecture from its embodiment at the Convex Exemplar to the HP V-class when they were acquired by HP. The HP Scalable Computer Architecture was proposed to extend the scalability of the V-class through a cross-bar to link four V-class nodes together in a large SMP machine. The V-class presents both local private memory (e.g., kernel) and a shared global memory (e.g., user applications).

Moving to and integrating with this complex architecture presented many problems, including existing hardwired hardware paths in the system configuration, lack of MP safety in existing semaphores, clock synchronization, TLB purging problems, cache coherency problems, and scalability.

Each problem had to be resolved without disturbing the existing architecture (software and hardware) too much, in order to integrate with existing solutions. For example, paths are used to reference resources in the HP-UX configuration. Previous configurations didn't include node IDs. While it is simple enough to add these, it is also important to still function when there is only one node. To solve the real-world system software engineering problem, relative paths were introduced, allowing existing node ID-less paths to function with node ID paths.

Similar solutions were used for other problems: software RPC to provide global TLB purge; clock drift software monitoring; limiting access to disallow simultaneous write and execute permissions on a page. Combined, these solutions allowed the architecture to meet existing product software requirements while forging ahead with hardware architectural enhancements.

Question: Have there been scalability studies of the system?
Answer: Yes, for aspects like locking granularity issues. We now have a much better understanding.

Question: How do HP-UX and Mach compare in performance, since Exemplar hardware is similar to V-class?
Answer: Mach was more scalable initially.

Question: What sort of applications is this system aimed at?
Answer: Best scalability is for scientific workloads; commercial workloads have less scalability right now.

SESSION: PERFORMANCE

STUB-CODE PERFORMANCE IS BECOMING IMPORTANT

Andreas Haeberlen and Jochen Liedtke, Karlsruhe University; Yoonho Park, IBM Watson; Lars Reuther, Dresden University; and Volkmar Uhlig, Karlsruhe University

Andreas Haeberlen and Jochen Liedtke presented research into the performance of stub code generated by the Flick compiler from the L4 IDL interfaces. Since the IDL compiler uses conservative knowledge of the interfaces, the stubs generated can have similar performance to cross-domain calls that must pass through the optimized L4 kernel.

To overcome these problems the compiler was modified to copy the stack directly and to use indirect string references (since the same address space is known). Doing so increases performance significantly and halves the size of the stub code (less marshaling).

The aim of this work is to feed back into the Flick compiler to support enhancements in more IDL primitives and to produce an order-of-magnitude improvement in intra-domain calling performance.

Question: Could I use the OS to make an addressing alias to avoid copying?
Answer: Yes, you can do this with L4, but this is not Linux semantics.

HP CALIPER: AN ARCHITECTURE FOR PERFORMANCE ANALYSIS TOOLS

Robert Hundt, HP

This work presented the Caliper performance analysis tool, which is being developed for upcoming IA-64 architecture systems such as Itanium. Caliper presents a comprehensive performance analysis tool to replace the limited tools being phased out and to support the complex functionality of the IA-64 architecture.

One of the biggest problems with existing tools is the need to recompile source code and relink or insert intrusive monitoring sequences. By this intrusion in the compilation or run-time process, such tools are used sparingly in large projects.

Caliper aims to overcome this by using support from the IA-64 processor to dynamically insert instrumentation (or in fact other types of program control/monitoring) callbacks into executing applications. These callbacks then call into a shared library to pass information to the front end. Doing so reduces the intrusion and improve performance, since monitoring code is only executed/added when needed.

The IA-64 architecture presents many complexities for such a tool. Currently the IA-64 implementation only supports a 25-bit branch, with 64-bit branch emulated, so callbacks have to be carefully integrated in order to overcome the emulation performance hit. Likewise, exceptions are complex in IA-64 (see the talk “C++ Exception Handling for IA-64,” below), which makes tracing C++ complex.

As a result of this approach, since only 12–40% of functions are reached, good performance can be had of between 1% and 80% overhead depending on workload. Performance monitoring, however, is only one of the possible uses of the tool; the hope is to extend its uses with support for pthreads, debugging, memory checking, and software fault injection.

Question: Do you need to stop threads for instruction/write updates on a bundle?
Answer: Currently yes, but we plan on developing an enhanced approach using templates to place breakpoints at the start of the bundle.

Question: IA-64 is very sensitive to code performance. How does this tool help?
Answer: Yes, compilers aren’t perfect, but they have improved over time. The output of Caliper can be used by the optimizer to improve performance.

Question: What is the effect on debugging tools?
Answer: We control the whole machine and effect execution. We want to add debugging facilities to our system to allow enhanced debugging too.

Question: How does this compare to DEC’s Atom?
Answer: I believe that was only static, not dynamic.

INVITED TALK

INTERACTION WITH TV IN 2003

Simon Gibbs, SONY Distributed Systems Lab

This talk and demonstration investigated the possibility for interactive television by the year 2003. Simon initially outlined the kind of system support we might have in 2003 to enable interactive TV services. In addition to the multi-channel content, such systems will have data connections in both directions. Return channels will use modems or broadband connections, depending on client cost.

With this environment, what form of interactive services might we see? A lot of services are enhancements to existing production facilities with multiple video feeds or data, such as sporting events, quizzes, news, etc. In such situations, existing data can be leveraged rather than making custom interactive TV productions.

During the talk the following potential service ideas were demonstrated in the context of a motor sport event:

- statistics that follow the cars’ positions, velocities, etc., and the ability to view a statistic of choice rather than being force-fed
- multiple-player interactive sport quizzes
- use of real car data to offer real competition in motor sport racing games
- superimposed racing of a computerized car against the real video footage, allowing competitive inter-

action and correct, realistic race car graphic integration

The promise of interactive TV has been with us for a while. This talk outlined the possible infrastructure and interaction ideas that may well find their way into your living room soon, if they can appeal to enough consumers.

REFEREED PAPERS: TOOLS

Incremental Linking on HP-UX

Dmitry Mikulin, Murali Vijayasundaram, and Loreena Wong, HP

This work covered a team's experience with providing incremental linking on HP-UX to improve link times in the development cycle of large applications. Incremental linking works by initially linking the application together and then being able to relink changes without completely relinking the binary.

Incremental linking has several problems. First, the padding areas must be correctly sized and placed to allow the best use of space for relinking. There may also be many symbols which are defined in several places and contexts (weak and strong symbols). Finally, changes required by the relink must be integrated and relocated as appropriate.

The approach taken in this work is to pad on a per function basis, with two copies (old and new) kept when relinking symbols to help resolve multiple symbols. The result is a linker capable of linking to produce a 3–11x performance increase, with a slightly slower initial link phase.

Question: Have you considered padding functions rather than object code padding?
Answer: Yes, HP-UX already puts functions in separate sections. It is a balance between padding and time saved, ultimately.

Question: What are the performance penalties of the approach?
Answer: Slower due to size increase and therefore increased cache misses, etc. But this is only used for development/debugging cycles.

Automatic Precompiled Headers: Speeding up C++ Application Build Times

Tara Krishnaswamy, HP

Tara Krishnaswamy introduced interesting work on the correct precompilation headers needed to speed up application time, since typically 50–95% of compilation time involves header processing in nightly builds. Some compilers (Microsoft) perform caching already, but they work on a per function basis, causing problems if dependence changes are made to avoid the header inclusion.

This work tries to overcome these problems by attempting to identify the initial part of each source file which is responsible for C preprocessor definitions. It then takes this region and precompiles it into a separate file. At build time, a checksum is used to determine whether the file has been updated. With no update the precompiled information is loaded into the compiler directly.

Problems exist in identifying this precompile region, since the C preprocessor has global scoping, and compilation flags can affect preprocessing too. These are overcome by identifying the configuration when precompiling and comparing configuration as well as checksums.

The result of this approach is a 20–80% speedup over normal compilation, at the cost of wasted space on processing duplicate inclusions. You must be careful when using version control systems, however, since the caches should not be shared and thus should not be in the control system.

Question: Can you use this to compile all sources?
Answer: Yes. We have a means to override, if needed. We haven't had any problems.

Question: Can you determine when a header is not needed?
Answer: No, you'd need a feedback-driven system.

C++ Exception Handling for IA-64

Christophe de Dinechin, HP

This talk covered the problems of implementing C++ exception handling for IA-64 architecture processors. C++ exceptions present several problems to solve in order to get good performance.

First, it is possible to throw any type, leaving the compiler to find the right implementation. Second, exceptions can be rethrown at runtime. Last, exception lifetimes aren't attached to the object scope. While these problems apply to all C++ compilers on IA-64, compiler optimization of these problems is important, since there is a 20x performance difference between -O0 and -O2 optimizations. Problems which the compiler must consider are: explicit parallelism, speculation (with alias problems), predication, register stack manipulations, memory state ordering, and register selection constraints. However, with exception processing, execution flow can take any code anywhere.

All told there is too much complexity to track all exception possibilities, so instead the compensation code is generated to give the compiler more freedom restoring visible state, copying registers, updating memos, etc., along with providing cleanup code to tidy up after exceptions.

The result is an exception implementation with little speed penalty, but with around a 30% overhead from cleanup and compensation code. This compares favorably to the PA-RISC implementation, but without the size overhead.

PANEL SESSION: SYSTEMS SOFTWARE RESEARCH AND TECHNOLOGY TRANSFER

Andrew Tanenbaum, Vrije Universiteit, Amsterdam; Rob Pike, Bell Laboratories; Marshall Kirk McKusick, author & consultant; and Rob Gingell, Sun Microsystems

Each panelist was asked to give his opinion on the subject.

RobG: Does it work? Yes and no. Social problems make it fail in the transfer step. Cooperation is difficult, research dies in the transfer. It is important to match timing and problem constraints and then be willing for it to take a long time.

Kirk: Open source is good for technology transfer, but the problem is money. There are many ways, not only the RedHat services approach. IBM, for example, seems to manage it well. In addition to transfer it can bring the costs down by an order of magnitude.

RobP: It's a catch-22 situation we have seen several times with Blit, UNIX, and Inferno. Success depends on being different from and the same as the market. You need to communicate, since colleagues will know more about work in companies other than your own. So you must use buzzwords to get attention and be prepared to transfer outside the company to get it back into the company again. Also, interns are a good way to transfer research.

Andy: Experience with Amoeba was not good – despite a book and a deal with a UNIX company, the product was expensive and no free trial was available. The UNIX company blamed the failure on giving it away. Given this experience the only way that seems to work is to transfer from research through students to companies.

Question: How to avoid research not looking too far out?

Andy: You should do what is innovative and see what comes out.

Kirk: Transfer takes a long time, so you should look far enough ahead.

Question: Are startups a good, quick path?

RobP: Companies don't understand how to keep employees from startups.

Question: Don't universities lack technology transfers?

WIESS in session

Andy: Not necessarily true. A lot of universities are doing good transfers, plus patents. But for some the tie is too close (e.g., UC Berkeley).
RobP: MIT gets a lot of their money from old patents.

Question: Is a good mechanism for transfer between companies to buy them?

RobP: Yes, but some are successful, or not. There are two reasons to buy: either to better the company or to stop the other company. The culture shift on acquisition is the biggest problem.

Question: Would Amoeba have transferred better if it had been given away?

Kirk: Yes.
Andy: I don't believe there is a future in free software. There's no free hardware or free books.

Question: Is shareware better?

Andy: I think this is just a marketing technique.
Kirk: Free software isn't really free. Integration is the key motivation too.

Question: We know what doesn't work. What does?

Andy: If you stay with the idea, e.g., Ethernet.
RobP: I agree, staying with the idea seems to work, e.g., C++. Luck is very important, or pigheadedness, but it takes a long time.
Kirk: You need a destination for the idea. The software development community (open source) is one such destination.

Question: How important is getting source into people's hands?

RobP: Very important. We get lots of UNIX folks these days. But UNIX wasn't free, it came from the community.

Question: Why are free UNIXes doing better at fracturing than commercial?

Andy: There are many free UNIXes! Linux is one man and nobody has tried to fracture it yet.
RobG: Linux is i386 and there was no other real i386 UNIX. The real test will come when new ideas are needed and seeing how they coordinate.

Question: Any good counter examples to go with?

RobP: Internal startups seem to work. Most transfer failures are the result of social problems.
RobG: They can work when research and ideas are not disruptive, but they need effort to transfer.

Question: If you could do a transfer again, what would you do?

Andy: Give Amoeba away.
RobP: Kill lawyers.
RobG: Help Rob Pike.

Conclusion

Andy: It seems that transfer from universities works best only when you can transfer it through students.

RobP: I agree with Andy; ideas move best with people and there must be a drive to succeed.

Kirk: Open source does work, but not in all areas.

RobG: Transfer can work depending on the disruption caused. Big ideas cause problems. Have realistic time frames.

INVITED TALK

Surfing Technology Curves

Steve Kleiman, Network Appliance, Inc.

In this talk, Steve gave an overview of the technology "waves" which Network Appliance saw and used, in order to further their business. Steve identified five particular waves:

- Filers – commodity storage appliances became possible. Standard components and protocols used in devices to achieve a small subset of reliable functions.
- Memory-to-memory interconnection and fail-over – commodity high availability support using dual ported disks and memory-to-memory interconnect to provide seamless fail-over.
- The Internet – cache appliances to move services to edge. Again, the aim of providing a simple set of reliable functions (Web and stream caching) in a server appliance.
- SnapMirror – traditional backup storage became too small or too slow for modern needs. But by using high density backup disks and fast data channels, good availability could be achieved.
- Local file sharing – direct access to storage through the VI Architecture using Fibre Channel, Infiniband, and the like. This enabled storage to be dissociated from machines, increasing scalability.

Each wave came as the possibilities of software and hardware brought new ways of looking at traditional problems, such as the move to appliances rather than monolithic systems.

In order to respond to these changes the structure of the system software needed to compensate. For example, no longer were general-purpose operating systems the slow choice. Instead, in order to get really good performance, specialized or refined system software – e.g., the DataOnTap architecture – is much more appropriate.

This led to the use of a small-message-passing operating system with no preemptive scheduling in Network Appliance. This allowed for low latency, high bandwidth operations with application-controlled resource allocation.

Question: Are there any problems coming up when we have a 10 Gigabit Ethernet?
Answer: No, there will be many small disks to distribute the load. I don't think it will really affect the architecture of storage servers.

Question: Are disks going to run out?
Answer: No. This has been a prophecy for ages. I don't think it will happen.

Question: Does memory speed scale?
Answer: It seems bandwidth is okay, but perhaps latency will be a problem.