THE VOV STORY; TCL IN EDA AND RELIGIOUS WARS

Andrea Casotto

Tcl Conference, Houston, October 17 2018

Altair

ABOUT ME

- 22 years as founder @ Runtime Design Automation
  - Obsessive efficiency

RUNTIME

- Chief Scientist for HPC @ Altair
  - By acquisition Sep 2017

Altair

MY TRAINING

- Learned programming from colleagues
  - 1985-1991 UC Berkeley CAD Group
- Ph.D. in 1991
  - Dissertation on “Automated Design Management”
  - **A system called “VOV”**
- In 1991, was Octtools Manager at UCB
  - Inherited >1M lines of code
  - Learned about reading / fixing other people’s code
  - Developed a sense of ”habitable code”

TIMELINE

- 1991 – Ph.D. Dissertation
  - VOV with Tcl embedded
- 1992 – Siemens AG project
  - Start of the Tk version of VovConsole
- 1995 – Founding of Runtime Design Automation
  - Bootstrapped start-up
- 2017 – Acquisition by Altair

THE STORY OF VOV

- Origins at UC-Berkeley
- VOV becomes Multiple Products
  - How Tcl is used in VOV
- Scripting language controversies
  - Tcl Perl Python …

VOV is also the name of and Italian liqueur.

VOV AND TCL IN BERKELEY

- VOV Project started in late 1987
  - Goal: Automate chip design process
- TCL also started in 1987
  - https://www.tcl.tk/about/history.html

Evans Hall, in Berkeley, where Ousterhout's office was

WHY TCL?

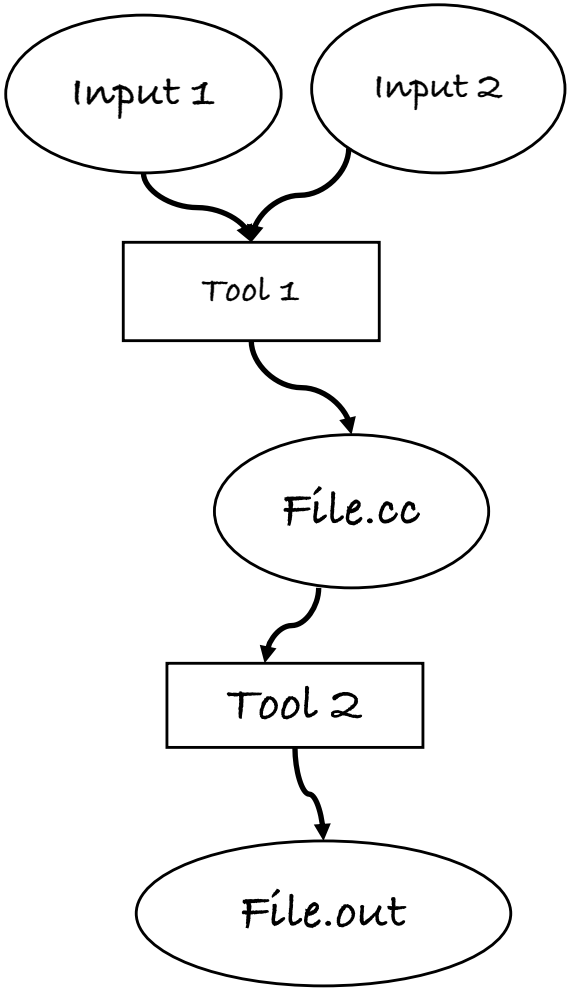
- John Ousterhout
  - Was my professor
  - A TA course on “how to teach”
- Ousterhout brand
  - Magic, Crystal, Sprite, …
- Elegance of language
  - All made sense
- Simple to integrate with C++

John Ousterhout

- Why do we still use it?
  - Amazing performance and capabilities

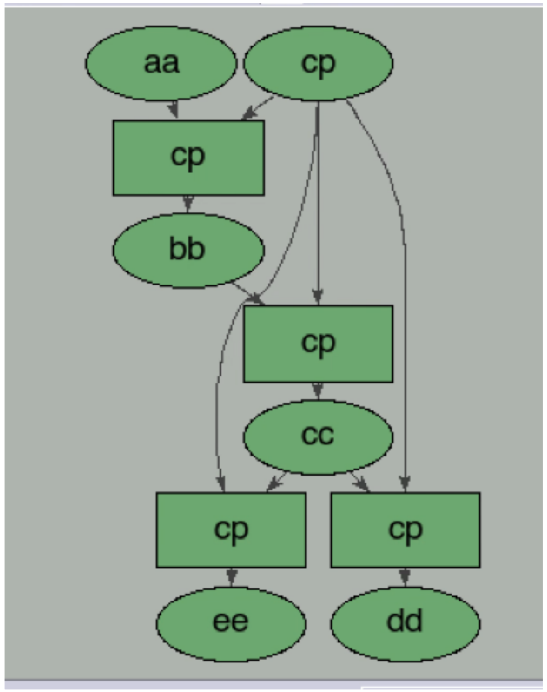
DRAWING FLOWS BY HAND

- I was TA in CS250 (prof Katz)
- Needed to teach students how to use Octtools
- Was drawing diagrams like this by hand
- Got tired
  - Needed a way to generate them automatically



WHAT IS A FLOW?

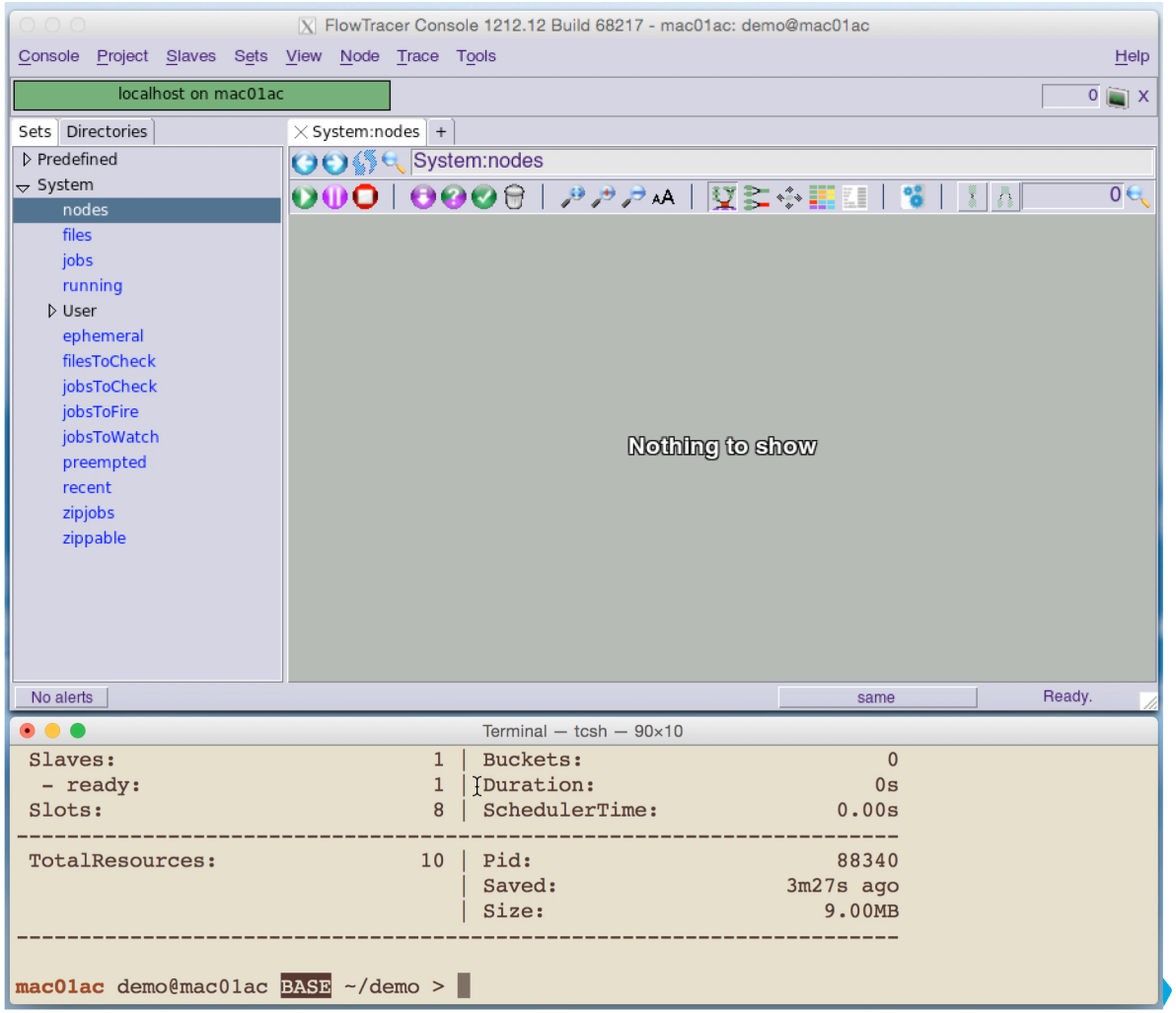
- A **Job** is a “Process” with inputs and outputs
  - Each output depends on all inputs
- A **Flow** is a collection of interdependent jobs



WHAT DOES VOV DO?

- Idea: Runtime Tracing
  - The tools **at runtime** inform VOV about their inputs and outputs
- Build a flow by simply executing tools

```
% vw cp aa bb
% vw cp bb cc
% vw cp cc dd
% vw cp cc ee
```



… AND YOU GOT A PH.D. FOR THAT?

- Most research on Flow Management at the time was exploring a "**normative approach**":
  - *"Thou shalt only do what the Flow Management System says"*

- Runtime Tracing supports **expert designers**
  1. Follow expert (record steps, trace the flow)
  2. Repeat when needed (retrace the flow)
- Traces became an important **training** tool for novices

MANY WAYS TO MAKE TOOLS TALK TO VOV

- **Encapsulation**:

  % vw cp aa bb

  Uses a Tcl script (a “capsule”) to capture I/O behavior based on command line

- **Instrumentation**:

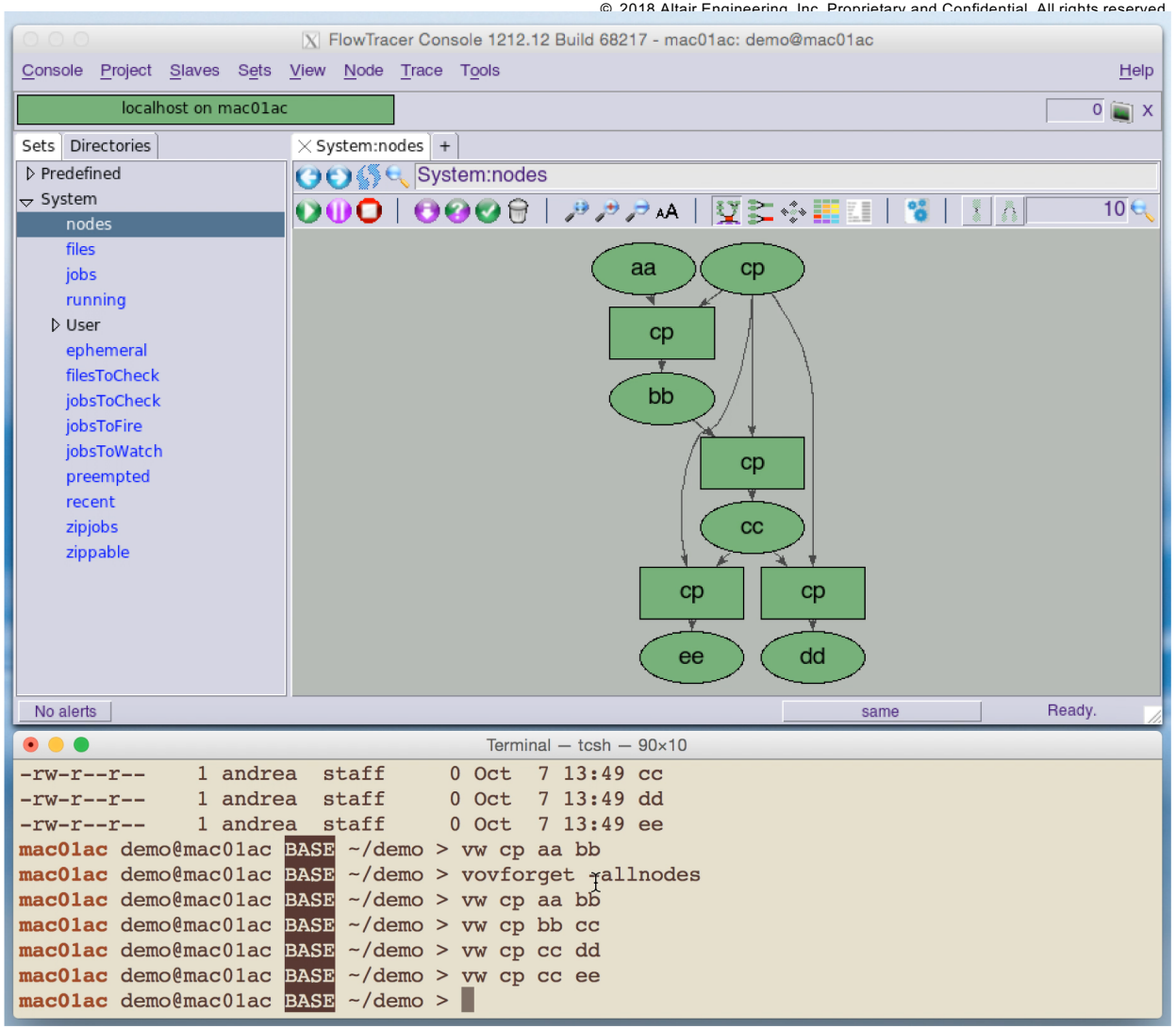
  ```
  #!/bin/csh -f
  VovInput $1
  VovOutput $2
  cat $1 |  tr –d foo > $2
  ```

- **Interception:**
  - of OS calls like open(), unlink(), mmap()
  - Uses LD_PRELOAD

FLOW DESCRIPTION LANGUAGE

- Build flows **without** executing tools
- Compact, simple

```
E BASE
X 2s
T vw cp aa bb
  I aa
  O bb
J vw cp bb cc
```



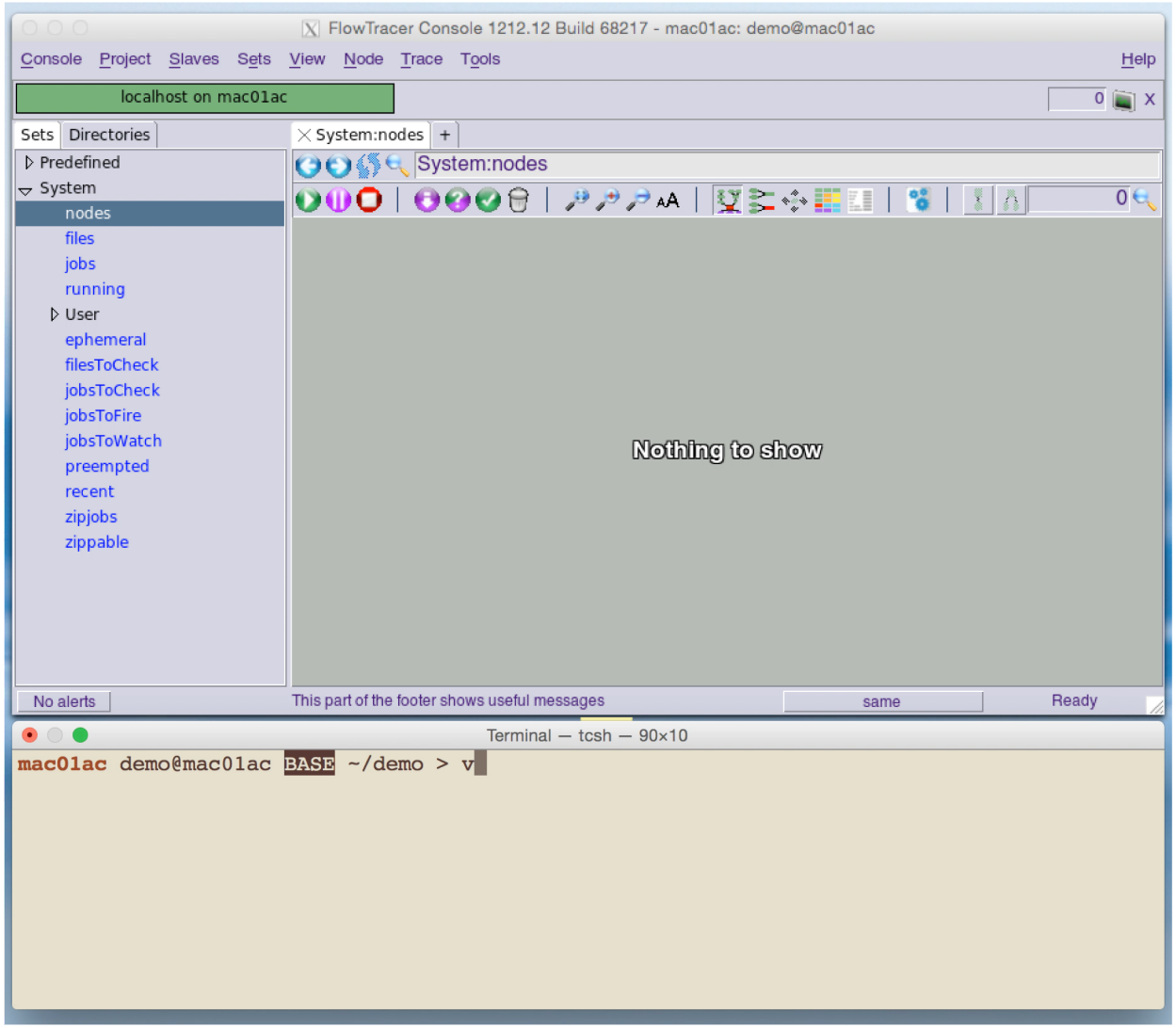
A MORE INTERESTING FLOW IN FDL

```
for { set i 0 } { $i < 30 } { incr i } {
    indir -create “subdir$i” {
        J vw cp ../aa bb
        J vw cp bb cc
        J vw cp cc dd
        J vw cp cc ee
    }
}
```

- Multiple directories
- Wide, parallel
- Hard to do with Makefile

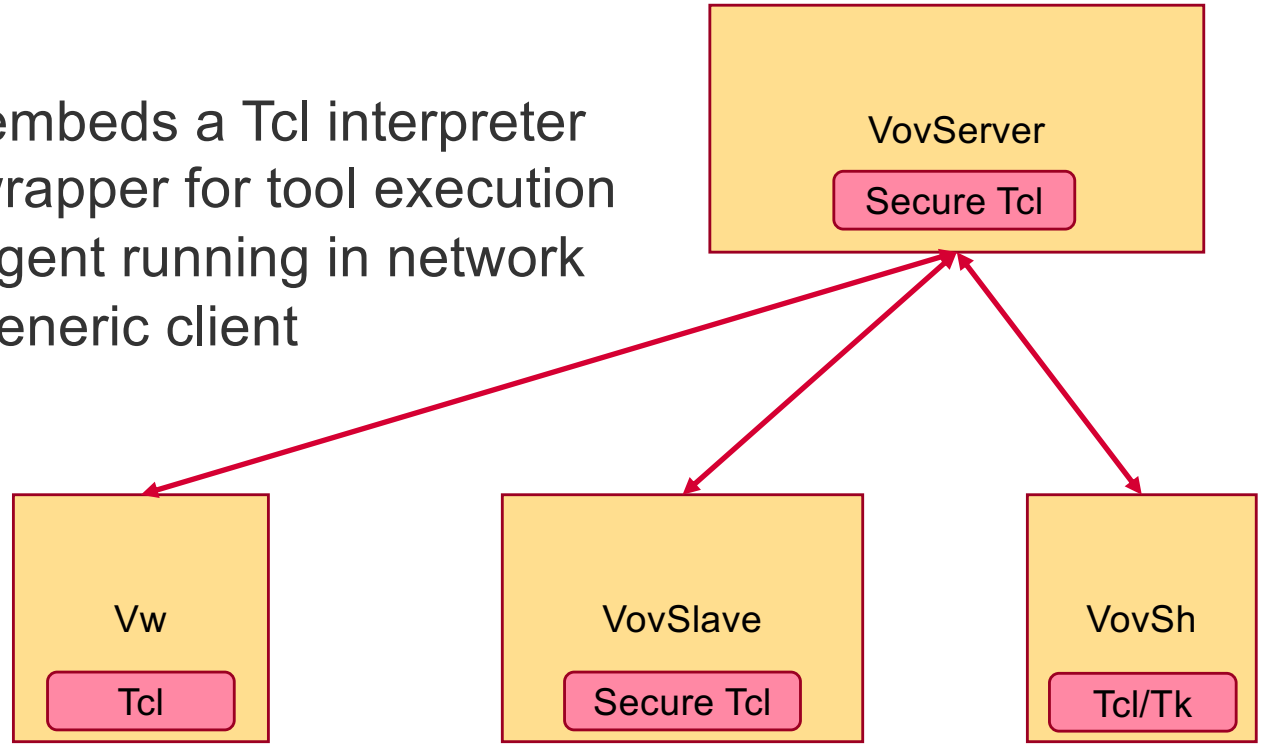
DEMO: A MORE INTERESTING FLOW

1. Build flow with vovbuild
2. Retrace different targets
3. Change an input
4. Retrace whole flow



ARCHITECTURE OF VOV

- Every component embeds a Tcl interpreter
  - vw – wrapper for tool execution
  - vovslave – agent running in network
  - vovsh – generic client



INTERESTING TCL EXTENSIONS - 1

- shift [list_ref]

```
while { $args != {} } {
    set arg [shift args]
    switch -- $arg {
        …
    }
}
```

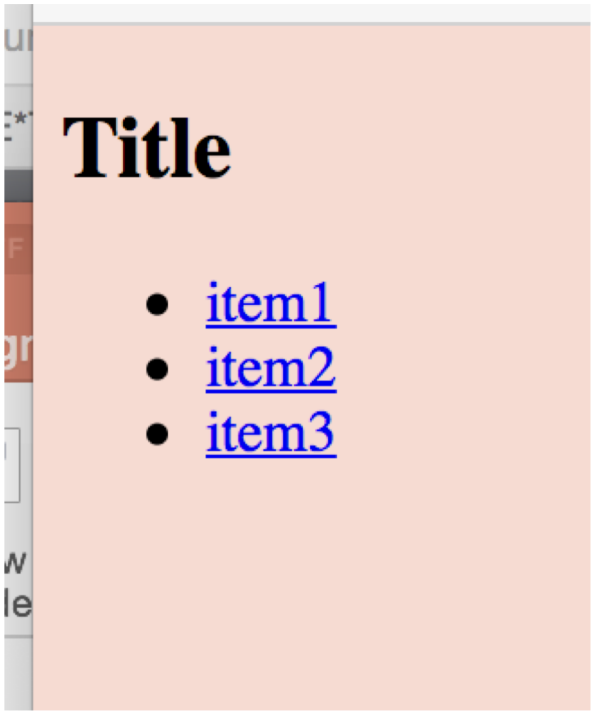
- indir {OPTIONS} {script}

```
indir –create subdir {
   J vw run_Simulation
}
```

INTERESTING TCL EXTENSIONS - 2

- HTML gen library

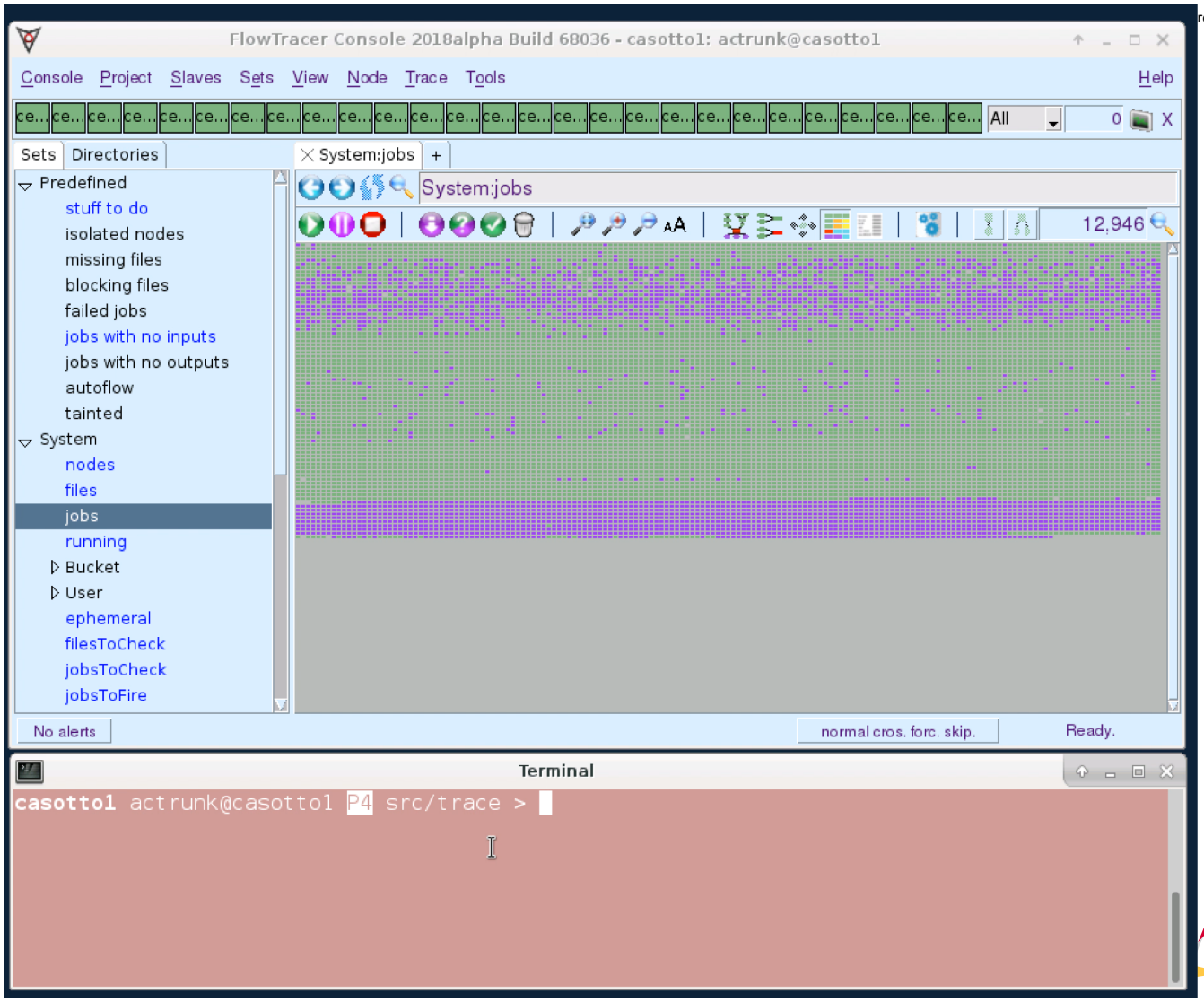
```
#!/bin/csh -f
# The rest is -*- Tcl -*- \
    exec vovsh -f $0 $*:q
…
VOVHTML_START
HTML {
  BODY {
    H2 { OUT "Title" }
    UL {
       set list [list item1 item2 item3]
       foreach elem $list {
          LI { HREF "/some/url/$elem" $elem }
       }
    }
  }
}
VOVHTML_FINISH
```



MILESTONE: 1996, VOV IS FASTER THAN MAKE!

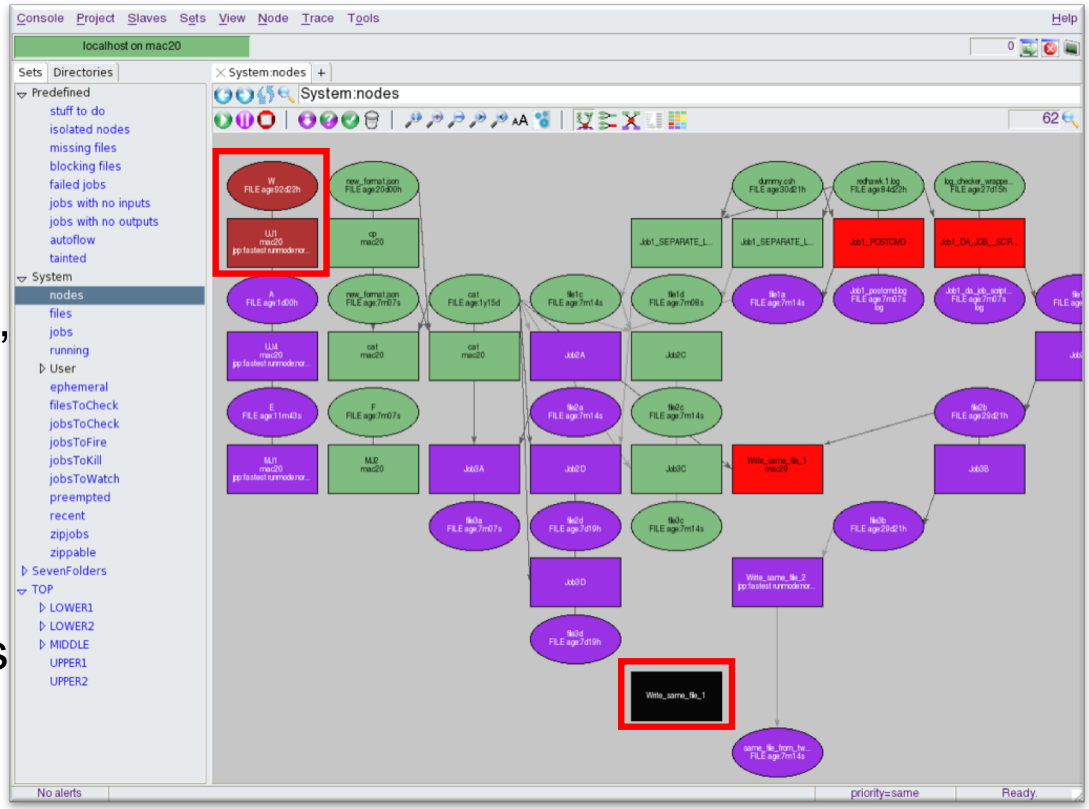
- VOV used to compile itself
  - Multi platform
    - Linux, solaris, windows
  - Multiple configurations
    - Debuggable, optimized, profile, coverage
- Unit testing, regression, docs, packaging

COMPILING VOV WITH 104 PROCESSORS



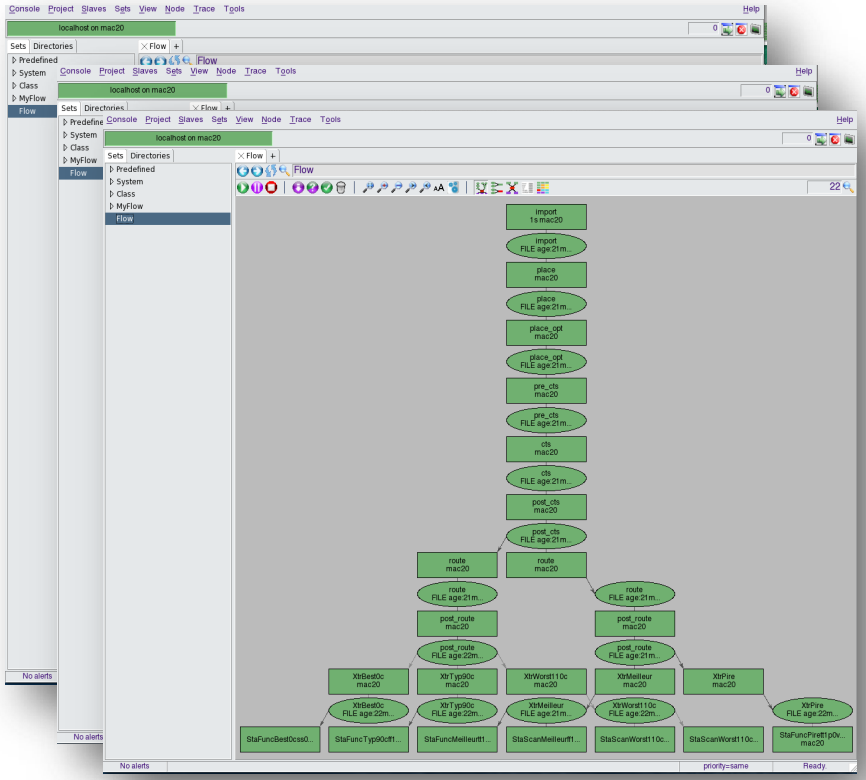
FLOW DEBUGGING

- Identifies flow problems
  - Missing files or dependencies
  - Flow conflicts
- Stricter file checking than “Makefile”
  - Job execution window
  - Timestamp consistency
- Saves designer time and resources



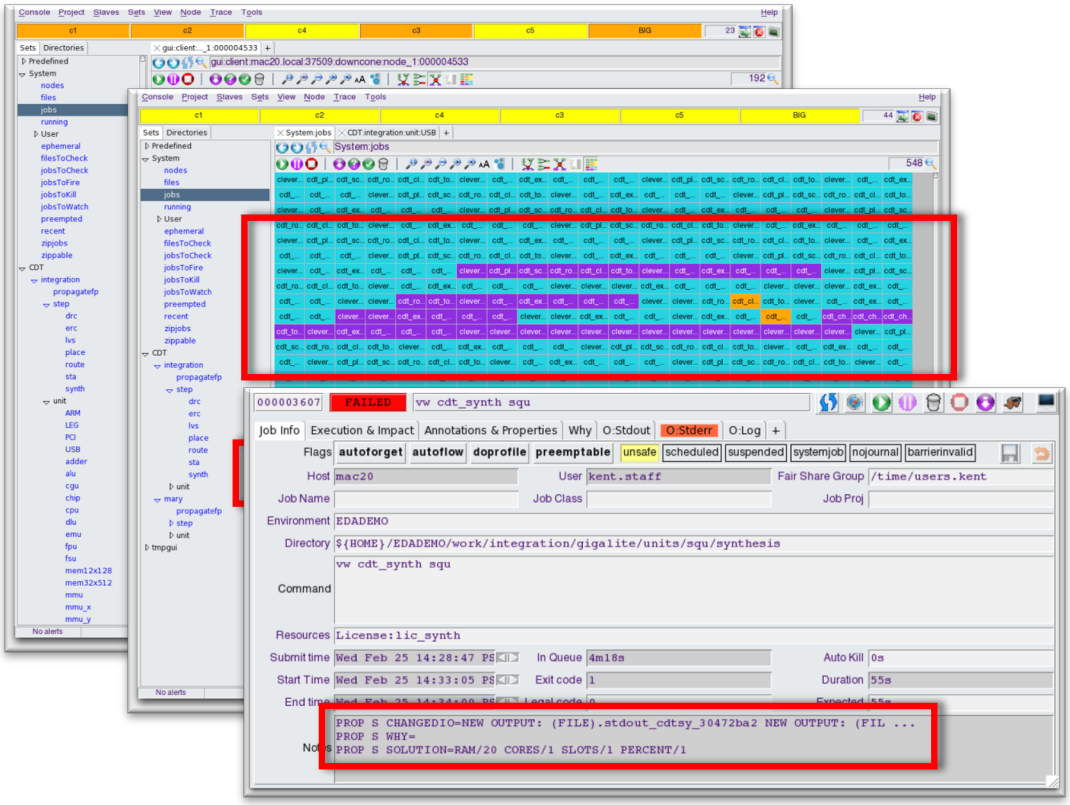
INCREASED PRODUCTIVITY

- Identify problems
- Make corrections or adjustment to the flow
- Continue without having to restart
- Better resource utilization



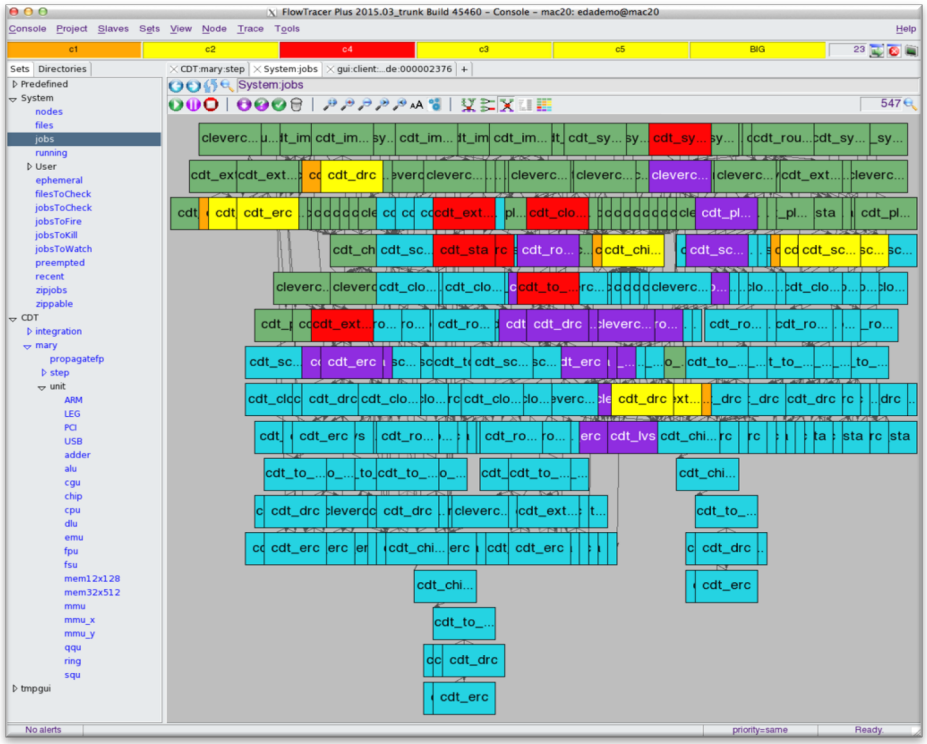
MORE VISIBILITY

- Easy to identify job status and failing jobs
- Quickly drill down for root cause analysis
- Full command line or web interface available



DESIGN ACCELERATION

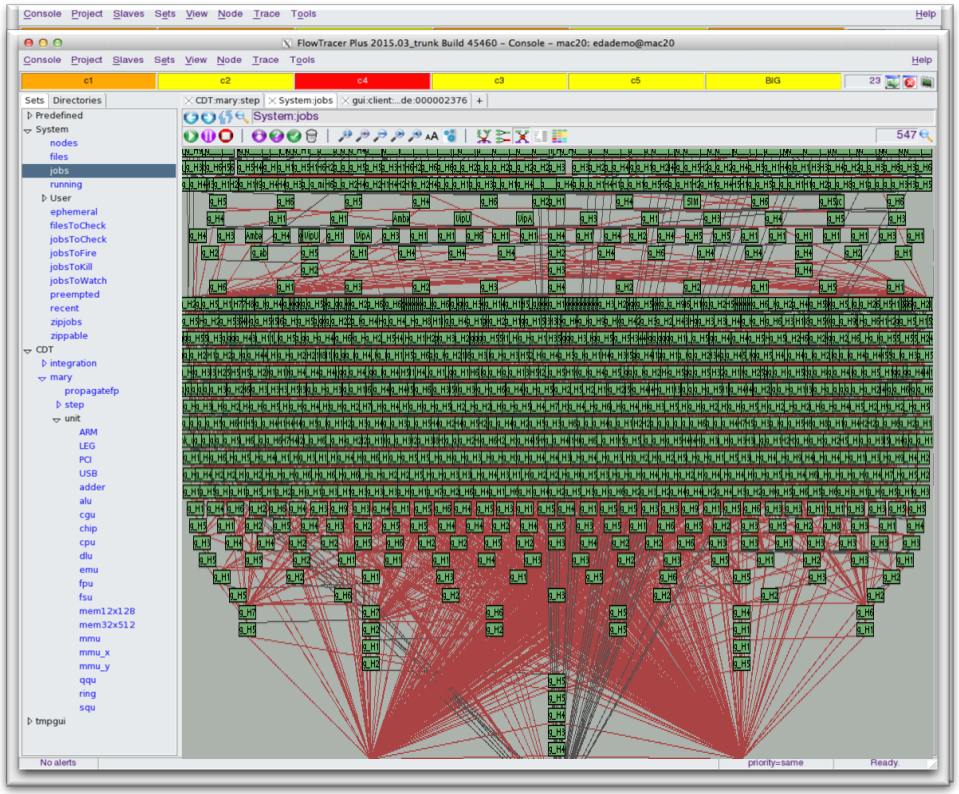
- Dependency Awareness
  - Jobs only start when all prior dependencies are met
  - Efficient parallel execution



- Maximizes resource utilization
  - Software licenses
  - Computer utilization

SCALABILITY

- Small memory footprint
  - 1M jobs → 1 to 2GB RAM
- Simple to complex flows
  - Scales from tens to millions of jobs
- Built-in high-performance scheduler
  - for faster throughput



OTHER CAPABILITIES

- Change Propagation Control
  - Stop insignificant changes (patented)
- Auto compression of large files
  - Based on flow info
- License / RAM / Cores requirements
- Built-in HTTP server
- Super extensible
  - Thanks to Tcl

MOST CAPABILITIES DEVELOPED FIRST IN TCL

- Examples:
  - Preemption of jobs
  - Distribution of licenses across geographies
  - Reconciliation of license usage
  - AWS monitoring, bursting to cloud

- After extensive testing and learning over many years …
  - Some capabilities are converted into C++
  - Tcl implementation is “executable specs” for the C++ implementation

IMPACT OF VOV IN EDA

- Most cell phones have a chip designed with VOV
- Largest flows: ~2 million jobs in a single flow (library validation flow)
- Typical chip design flow: 2k to 20k jobs

GROWING FAMILY OF PRODUCTS

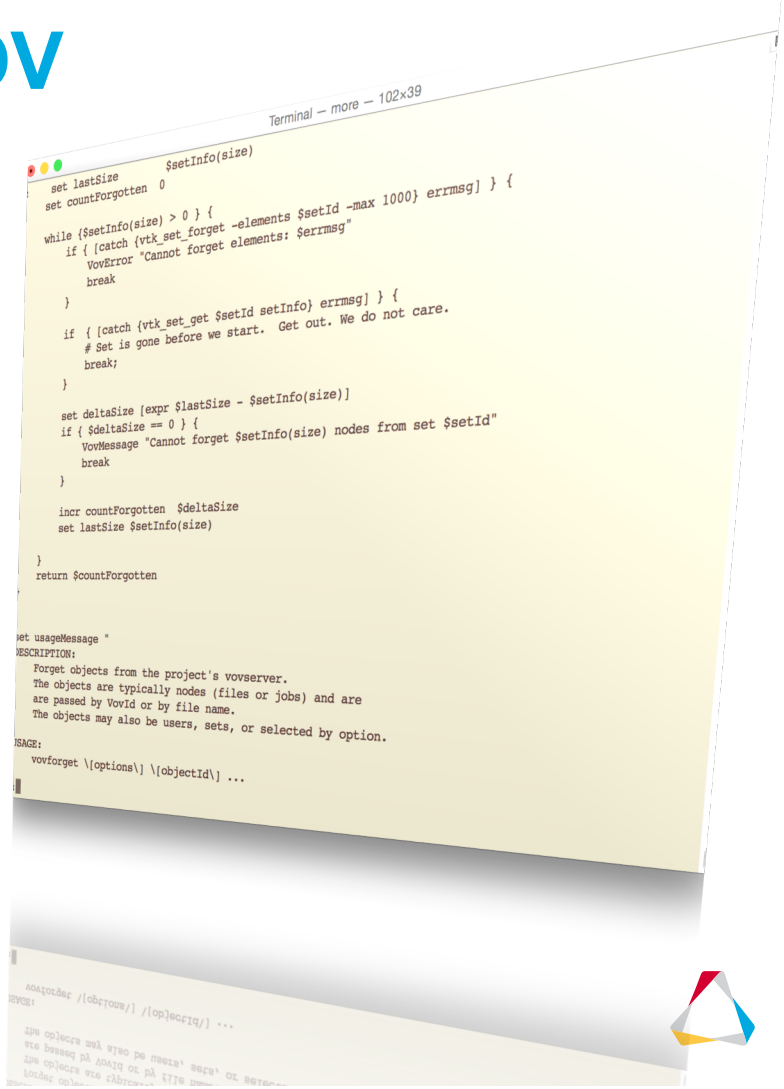
- In the beginning, there was FlowTracer
  - New name for Original VOV
  - Generic flows
- … begat NetworkComputer
  - Just the scheduler part
  - Specialized for many independent jobs
- … begat LicenseMonitor, LicenseAllocator, WorkloadAccelerator
  - License management layer in NC
  - License distribution across multiple sites
  - Hierarchical scheduling

MANY PRODUCTS: SAME CODE BASIS, DIFFERENT TCL "DRESSES"

LicenseMonitor™	LicenseAllocator™	WorkloadXelerator™	WorkloadXelerator Meta™	HERO™	FlowTracer™
• Software license monitoring & management • Optimizes license mix • Reduces license cost	• Multi-site license allocator • Real-time license adjustments • Visibility into license allocations & usage	• High-performance job scheduler • >5x faster than competition • Full featured: FairShare, preemption, reservations, etc.	• High-performance hierarchical scheduler • ~6-10x increased throughput • Scale-out strategy	• High-performance scheduler for hardware emulation • End-to-end solution: compilation, emulator selection, & regression	• Platform for developing, executing design flows • Reduces design risk & cost • Accelerates time to market

CURRENT COMPLEXITY OF VOV

- C, C++ code: ~360k lines
- Tcl/Tk:
  - .tcl 230k
  - .mod 22k
  - .cgi 62k
- Regression Testing:
  - *.tcl 45k lines
  - *.csh 50k lines



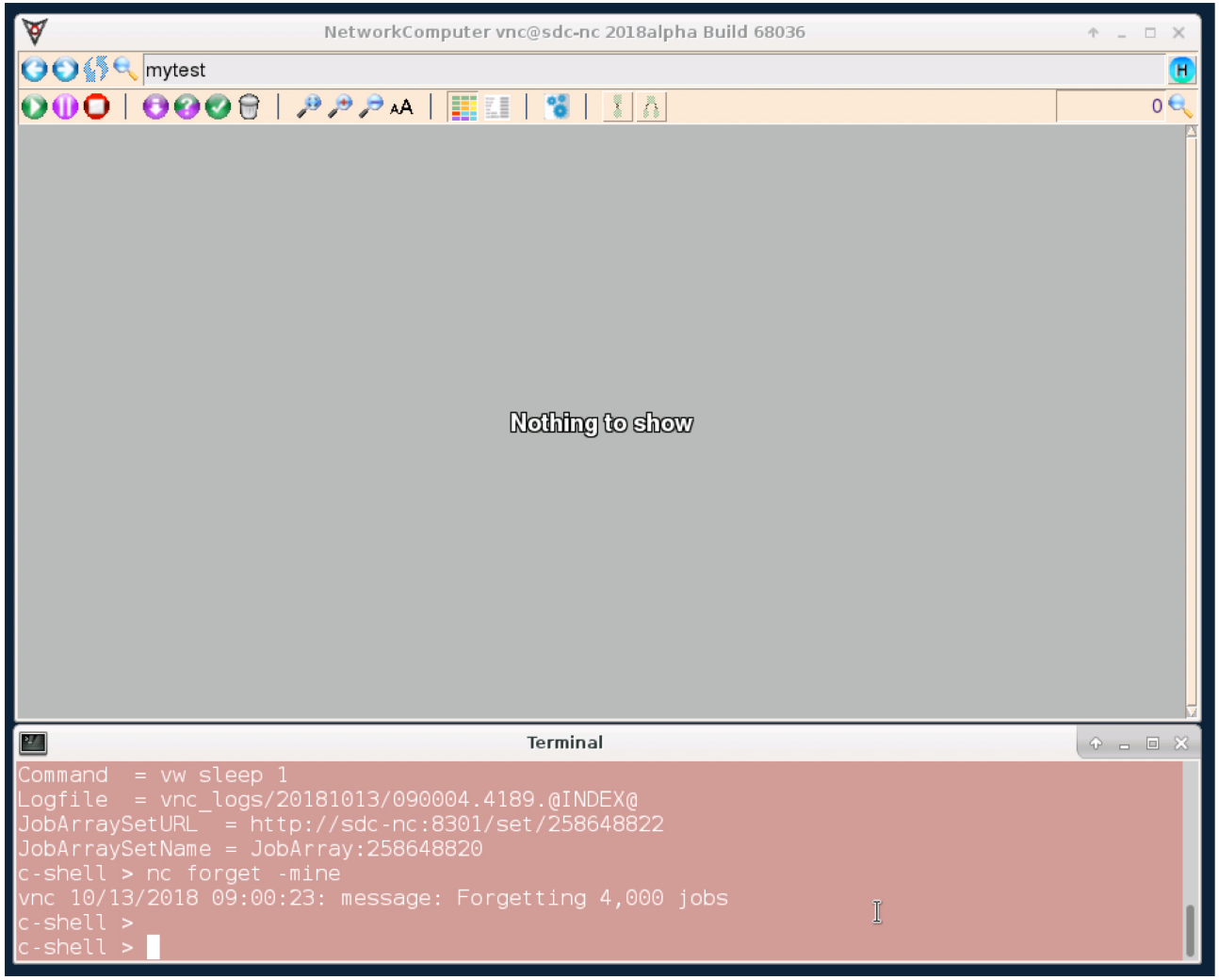
LICENSEMONITOR IN ONE SLIDE

- Real time license management
  - Identify **underutilized licenses**
  - Usage by project, business unit, user, ...
  - License expirations

- Greater operational efficiency
  - Decisions based on real time and historical reports
  - Greater management and designer visibility

NETWORKCOMPUTER: FASTEST JOB SCHEDULER*

- Event driven scheduler
- Small memory footprint
- Full-featured
- Speed record:
  - 10,000 sleep 0 jobs in 8s



* Vs. Slurm, SGE, LSF, …

TCL API IN VOV

- Total of 410 “vtk” (Vov Tool Kit) procedures
- Example: For Reservation
  - `vtk_reservation_create <type> <what> <q> <start> <end>`
  - `vtk_reservation_get <id> <array>`
  - `vtk_reservation_update <id> <fieldname> <value>`
  - `vtk_reservation_delete <id>`

- Attempts to use SWIG to port to multiple languages
  - have not yielded results

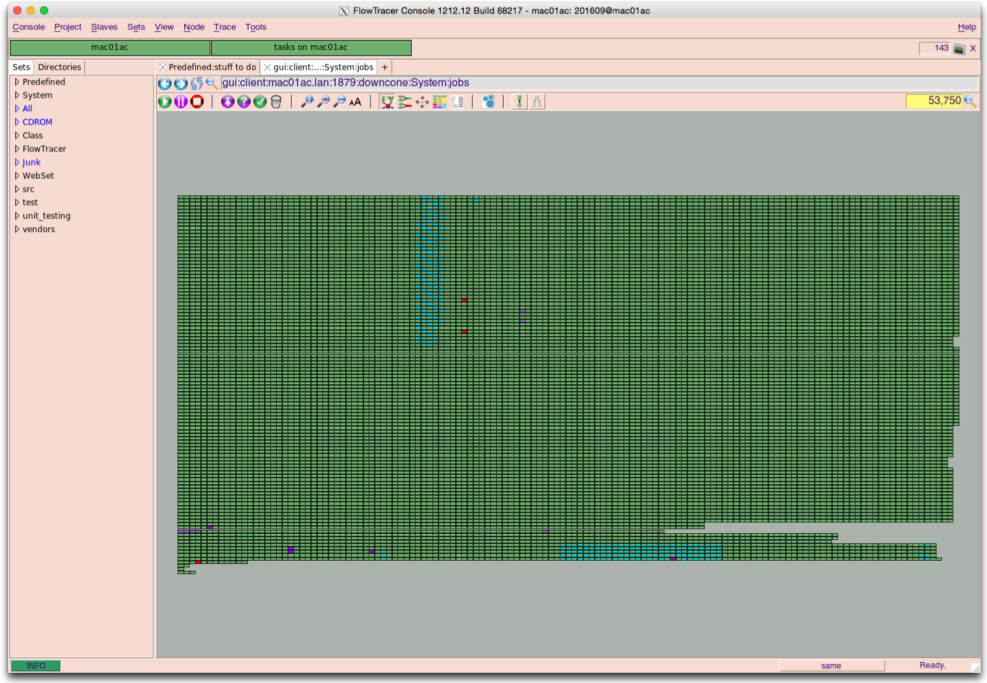
SCRIPTING LANGUAGE CONTROVERSIES

IS TCL POPULAR?

- Tcl not in list of most popular languages
  - Python is popular
  - Perl less and less
- Yet still Tcl is a **de-facto standard** in EDA
  - https://www.cadence.com/content/cadence-www/global/en_US/home/training/all-courses/82158.html
- And also used in mechanical CAD

CUSTOMERS' REACTIONS TO TCL/TK

- About Tcl
  - "yuk!"
  - "Why don't you have a Perl interface?"
  - "Why don't you have a Python interface?"
  - "Why don't you have a REST API?"
- About Tk
  - "It looks like it is from 1980!"



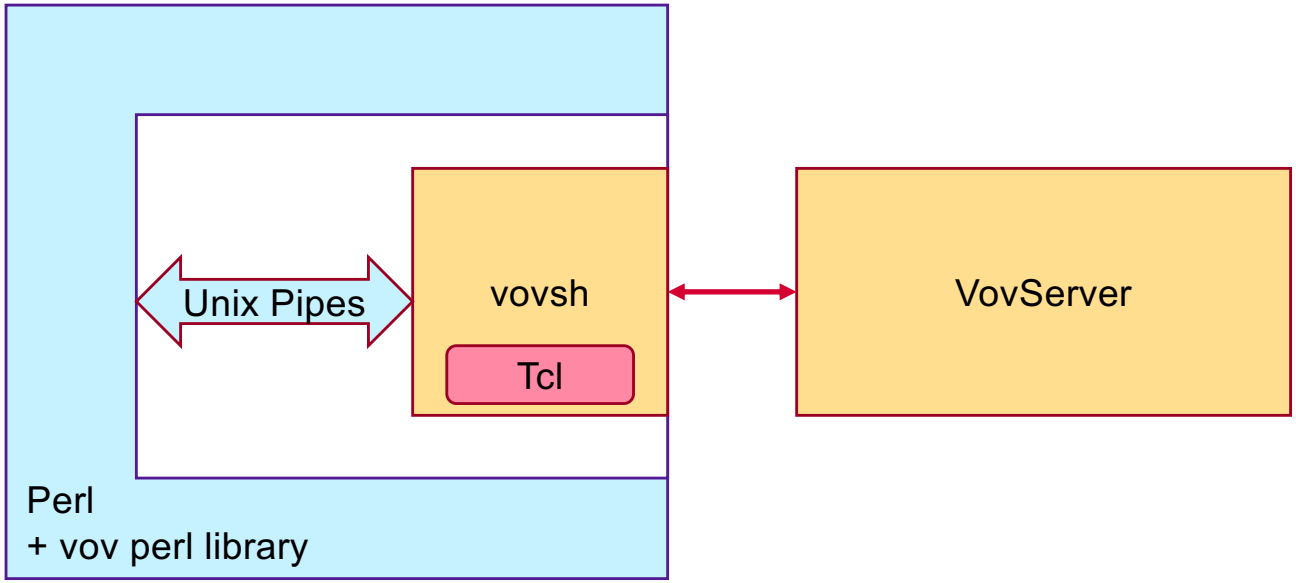
CONTROVERSY HAS TASTE OF ”RELIGION”

TCL OVER PERL: HIGH-LIGHTS FROM HTTPS://WIKI.TCL.TK/1330

- Tcl is simpler. Those without a C/Unix background generally find Tcl syntax far easier to learn and retain.
- Tcl is smaller.
- Tcl is easier to extend, embed, and customize.
- Tcl source code traditionally is a model of lucidity. Perl source code traditionally is dense in magic.
- Tcl/Tk is far more portable than Perl/Tk, and generally more current.
- TCP networking is more succinct and less intimidating.
- Tcl's exec, open and socket are gems of accessible and portable functionality, in comparison to the analogous Perl offerings.
- Tcl's unified channel API makes life much easier, particularly on Windows.
- As of spring 2001, Tcl's Unicode [1] capabilities are considerably more mature.
- As of spring 2001, Tcl's threading savvy (read "Tcl and threads") is considerably more mature.
- Subjective stuff: some people find Tcl a better fit to their own sensibilities.
- You can read your own code 6 months after you've forgotten how the program worked.
- (file)event, trace and friends often solve requirements for functionality better than threads.
- Tcl is *way* ahead of Perl in VFS capabiltities; fuse provides an example of the potential consequences.
- As "Tcl's string handling has been written by paranoiacs", to quote DKF, Tcl is immune to many "format string vulnerabilities".
- Yerlp: this page has been here for years, without adequate emphasis on "The uniqueness of safe interps".
- Deployment of Tcl does not need a full installation. It can be delivered as starkit or starpack

IN THE END, CUSTOMER IS ALWAYS RIGHT

- Best implementation
  - A kludge!
- Then demand subsided around 2008 … 2010



TCL VS PYTHON

- **“What is the best programming language to learn first?”**
  - **Python** is ranked 1st while **Tcl** is ranked 29th. …
  - Source: https://www.slant.co/versus/110/5079/~python_vs_tcl
- We still have not done a Python port

REST API: THE NEW DEMAND

- Apparently, cannot enter the Fin-Tech market without it
- Not really a standard
- We are doing it!
- Also we have Python wrapper for REST API

```
# Tcl equivalent, w/o error checking
vtk_node_get 029462592 nodeInfo
puts $nodeInfo(status)
```

API to retrieve status of a job

```
http://HOST:PORT/api/v1/jobs/029462592/status
```

JSON reply


```
{
  "startrow":1, "endrow":1,
  "query":"SELECT status FROM jobs WHERE
idint==29462592", "errormsg":"",
   "columns": [ {"col":0,"field":"status"}
], "rows":[ ["VALID"] ]
}
```

MOST IRRITATING THING IN TCL

```
# This example inspired by https://news.ycombinator.com/item?id=4921066
proc ciao {} {
   # A comment with }
   puts "Ciao"
}
ciao
```

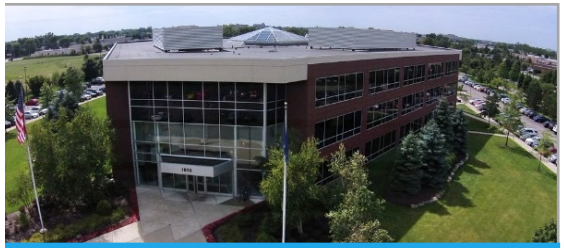
```
% tclsh ./badtcl.tcl
Ciao
invalid command name "}"
    while executing
"}"
    (file "./badtcl.tcl" line 4)
```

OPEN ISSUES

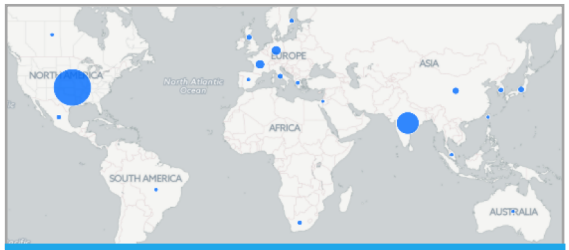
- Lots of dead Tcl code
  - No code coverage tool
  - Some coverage at “procedure level”

- Tcl Interpreter initialization dominates runtime
  - “nc run hostname” 95% of time is spent initializing Tcl

In 2017, ALTAIR acquires Runtime Design Automation



Founded **1985**
Headquartered in Troy, MI US



71 offices
in 24 countries

$333M
2017 Revenue

50+
ISV partners under our unique, patented licensing model

2000+
Engineers, scientists and creative thinkers

5000+
Customer installations globally

60,000+
Users

Altair

SUMMARY

- Tcl choice in the beginning was opportunistic
  - Largely a UC Berkeley connection
- Tcl has been instrumental in the success of Runtime Design Automation
  - Several users of Tcl also in Altair
- VOV development continues at Altair
  - **Tcl still actively used** to increment functionality