ce customer sites. I also
Command (REDCOM)
onsor of our project. And
in Germany two times.
evelop the Joint Exercise
d on the Model Interface
project took my Scenario
o specification languages
system. Maybe someone
r validated my approach.
h human and computer
hanged easier, are more
or another project. The
ge since, after what I had

oland & Associates, the
Program for JTLS and
Monterey to make JTLS
RISC machine. There
and the VAX that was
ber what that difference
y Roland flew me there

CHAPTER 20

1985 - 1989

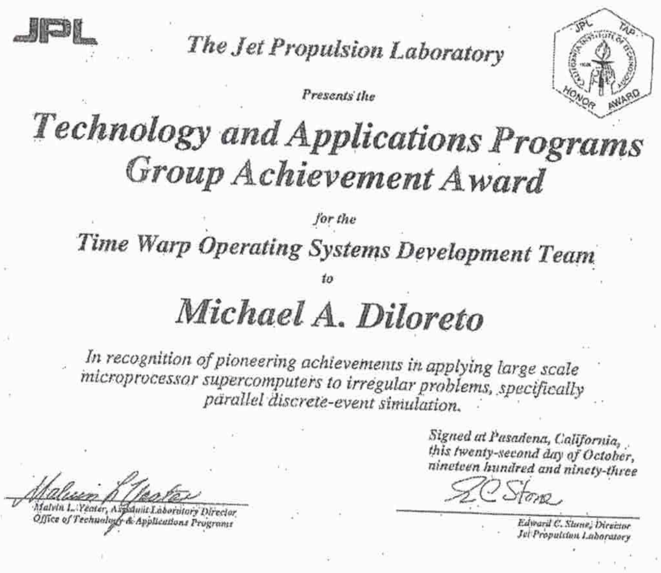
JPL - Time Warp Operating System

AFTER JTLS, I stayed on at JPL in the same building, but on a new project: *Timewarp on the Hypercube.* Really. That's what it was called. *Timewarp* was the name of an operating system for Distributed Discrete Event Simulation, and the *Hypercube* was a large-scale distributed multiprocessor. The Time Warp Operating System (TWOS) was invented by Dave Jefferson, a Computer Science professor at UCLA. The Hypercube Parallel Processing system was built by Caltech. The project was funded by the U.S. Military to potentially do more war games.

Applications running under the Time Warp Operating System were Object-Oriented and coded in C. The operating model was similar to Smalltalk in that it worked by objects sending messages to each other or to themselves. It was not so much a programming language like C++, which wouldn't be available for a few more years. The way this worked for Discrete Event Simulations was that the messages were stamped with the simulation time. In the Simscript language I described earlier, the equivalent notion was called scheduling an event. TWOS ordered the messages by simulation time and then executed the destination object's code for each message, advancing the simulation time for each one.

When I joined the project, there were 4 or 5 people already working on it. It was not in very good shape, though. Each person had his own version of TimeWarp, and none of them worked. The computer we had to use was that same VAX 11/780 in the JPL Graphics Lab. Software organization is one of my strengths. That's not normally mentioned as an attribute of programming, but I think it's important. I kind of took over and made one good version of the software and got rid of all the rest. Then everyone used the version I controlled. I was not the boss or the technical leader of the project. Nobody had to obey me. It's just that I was doing something useful.

An interesting thing we did as a project was to have regular configuration control board meetings. These were run by the manager and everyone was included. Normally I would not like something like that but, in this case, it was simple and effective. The idea is that the "configuration" of the software is basically what's in it, what features does it have, and so forth, and that there should be some control over it. That is, there should be some control over what new features are added to the software, when and how. In the meetings, anyone could make a suggestion for a new feature or some change to the software. Then everyone in the meeting could discuss it. It just kind of worked out, since I was personally in charge of the software, that I was the best one to evaluate the proposals. I could judge not so much whether the change was a good idea or not, but what kind of effect it would have on the system and how difficult it would be to do. I'm naturally receptive to new ideas, so there was never really any dispute, although I'm also naturally pretty strict about the integrity of the software. Usually I'd be the one to implement the changes anyway, so I was always able to make sure they were okay.


JPL

The Jet Propulsion Laboratory

Presents the

Technology and Applications Programs Group Achievement Award

for the

Time Warp Operating Systems Development Team

to

Michael A. Diloreto

In recognition of pioneering achievements in applying large scale microprocessor supercomputers to irregular problems, specifically parallel discrete-event simulation.

Signed at Pasadena, California, this twenty-second day of October, nineteen hundred and ninety-three

[illegible]
Malvin L. Yeater, Assistant Laboratory Director
Office of Technology & Applications Programs

[illegible]
Edward C. Stone, Director
Jet Propulsion Laboratory

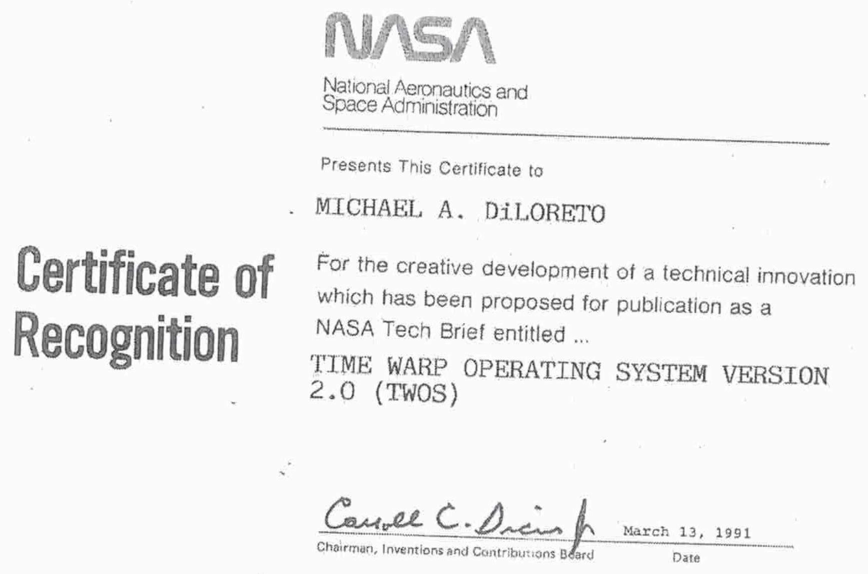
We were doing pr
Then we got a real Hy
The way they're conn
a cube, but with more
directly to every other
of the hypercube to
be routed through so
actually making it wo

There was a separa
that was supposed to
was not used on the
operating system was
had been going on fo
the Summers, a prof
Florida State Univers
was Steve Bellenot. H
and we worked togeth

Certificate o
Recognition

We were doing pretty well with TimeWarp running on the VAX. Then we got a real Hypercube. It was made of 32 Intel 8086 processors. The way they're connected together is the hypercube topology. It's like a cube, but with more dimensions. Processing nodes are not connected directly to every other node. To get a message from a node at one corner of the hypercube to another one at another corner, it would have to be routed through some other nodes. It's basically a simple idea, but actually making it work can be pretty complicated.

There was a separate part of the software called the Machine Interface that was supposed to deal directly with the 8086 microprocessors. That was not used on the VAX, so even though the TimeWarp part of the operating system was kind of okay, the 8086 part was not. The project had been going on for at least a couple of years before I got there. In the Summers, a professor of Computer Science and Mathematics at Florida State University came to JPL to work on the project. His name was Steve Bellenot. He came for the Summer again when I was there, and we worked together on making TimeWarp work on the Hypercube.



NASA

National Aeronautics and Space Administration

Presents This Certificate to

MICHAEL A. DiLORETO

Certificate of Recognition

For the creative development of a technical innovation which has been proposed for publication as a NASA Tech Brief entitled ...

TIME WARP OPERATING SYSTEM VERSION 2.0 (TWOS)

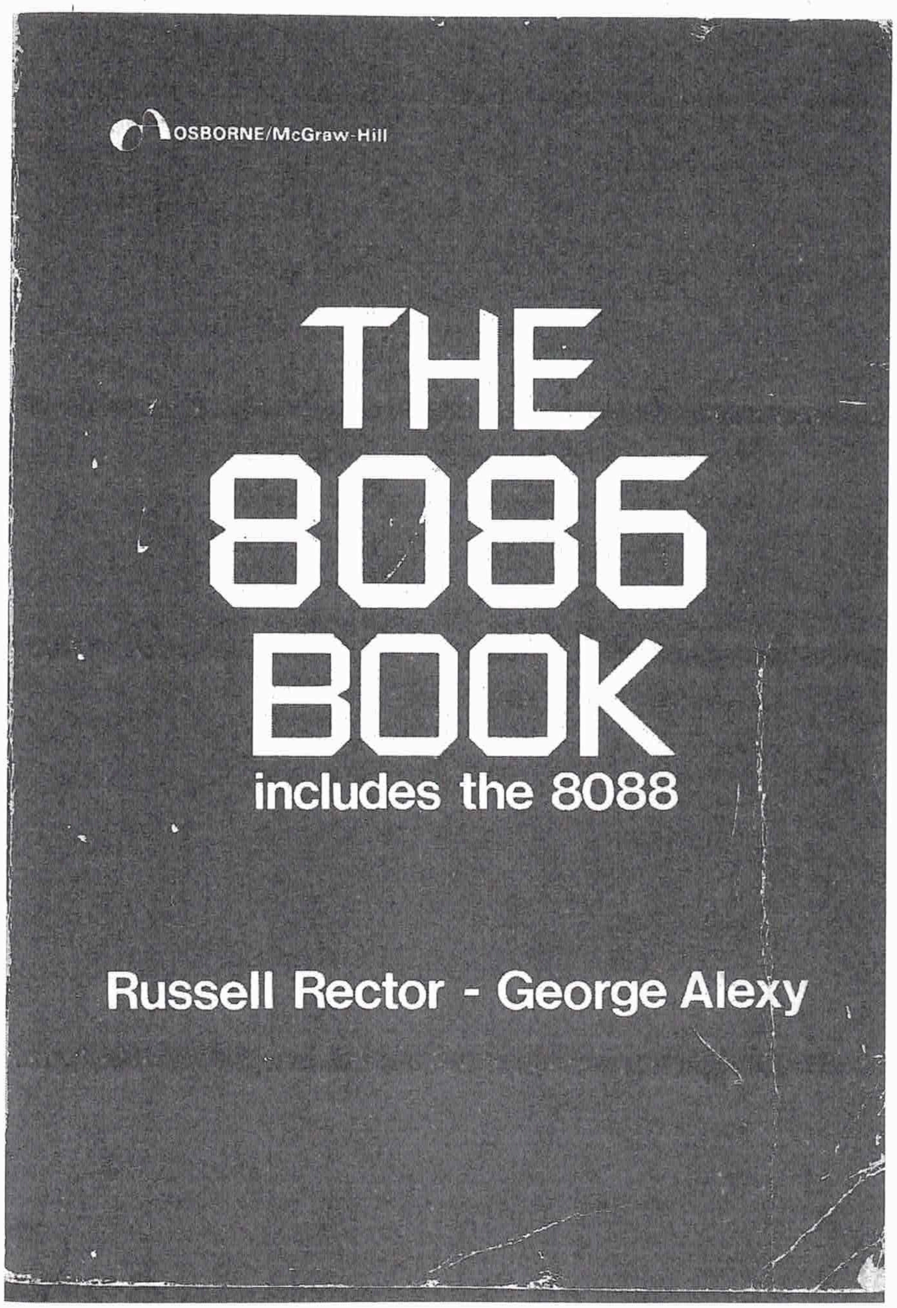
[illegible] March 13, 1991

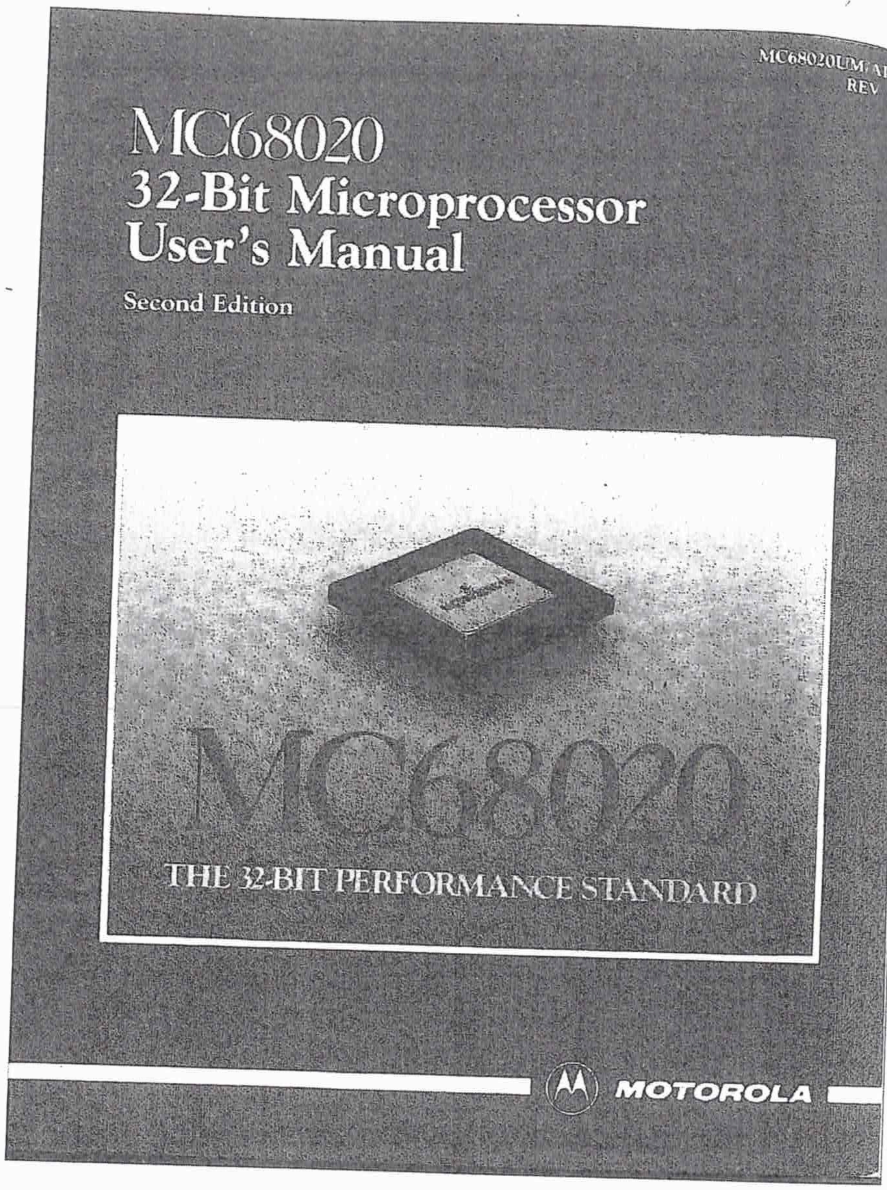
Chairman, Inventions and Contributions Board Date

By now, the contract deadline for the project was that Fall. If TimeWarp was not working on the Hypercube by then, the project would be canceled. Steve and I worked on it every day for 2 or 3 months that Summer, 1986 I think. Steve had been on the project longer, and he understood the theory of TimeWarp better than me. I happened to be an expert in Intel processors and parts, and really good at debugging. We were a good team. We complemented each other's strengths.

I'm pretty innovative too. I found several VT-100 terminals, maybe 4 or 5 of them, and connected them to some of the Hypercube nodes. That way, the software could print things and we could even get into the debugger. Sometimes software is so bad that it would have been faster to rewrite it than to debug it. The trouble is you don't know that until you've been debugging for awhile, and then you think you're almost there. The Machine Interface code was like that. It took the two of us a couple of months to get it to work. But we did it. It worked, there was a demo, and the project was renewed for another year. It would actually go on for several years more.

was that Fall. If
then, the project
for 2 or 3 months
roject longer, and
ne. I happened to
ood at debugging.
s strengths.
terminals, maybe
Hypercube nodes.
d even get into the
d have been faster
't know that until
ink you're almost
ok the two of us a
orked, there was a
It would actually





MC68020UM/AD
REV 1

MC68020
32-Bit Microprocessor
User's Manual

Second Edition

MC68020

THE 32-BIT PERFORMANCE STANDARD

MOTOROLA

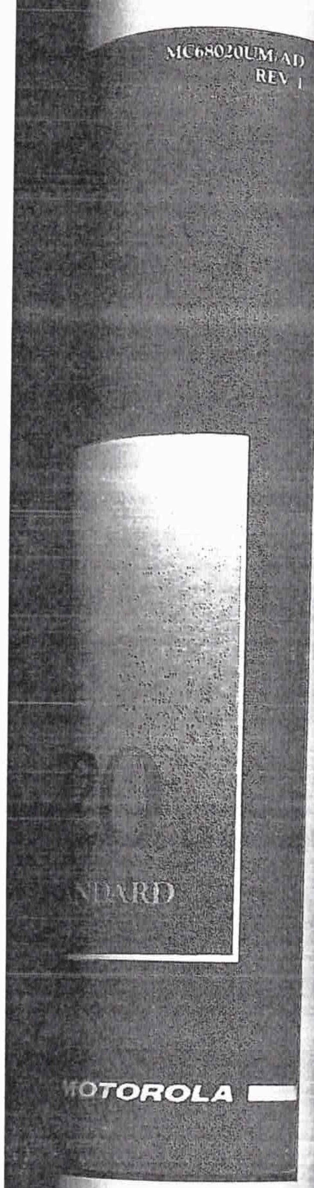
Caltech built a Mark III Hypercube that was 256 Motorola 68020 processors, two per each node. Brian Beckman was the real technical lead of the project. He had a PhD in Astrophysics from Princeton, and he was a bonafide genius, to me he was anyway. We were in the so-called Foothill offices of JPL in Pasadena, and the Mark III Hypercube was installed at the main Lab. He suggested we could make TimeWarp work on that much

bigger and bett
to it. So, we we
work. I happen
announced the
left out. But we
for our project.
and we never u

Our group
Each one of u
that was as b
over punched
used our local
multiprocesso

The Time
for a little bit
context switch
RISC machin
think the inst
it as RISC wa
cycle. The Mc
Computer). T
cycles on vari
same reasons
according to
instructions t

The Sun
mind. C has
for holding l
so instructic
instructions
The SPARC
registers anc
references t
go faster. V
mapped to



256 Motorola 68020
the real technical lead
Princeton, and he was
the so-called Foothill
cube was installed at
p work on that much

bigger and better Hypercube, and he knew the people to give us access to it. So, we went there on a weekend, and working together we made it work. I happened to know the Motorola 68000 already. On Monday, we announced the good news, and the manager was mad. He didn't like being left out. But we didn't get fired because it really was a giant step forward for our project. Now we could access the Mark III Hypercube remotely, and we never used the Mark II Hypercube, the 8086 one, again.

Our group was maybe the first one at JPL to get Sun Workstations. Each one of us had our own Sun 3 Workstation on our desk. To me, that was as big an advance in technology as timesharing had been over punched cards, maybe bigger. I made a version of TimeWarp that used our local network of Suns, which included a Sun 4 SPARC, as a multiprocessor system.

The Time Warp Operating System was written entirely in C, except for a little bit of Assembler Language code for each type of machine for context switching. That included the Sun SPARC, which was an advanced RISC machine. The machine actually had a lot of instructions, so I didn't think the instruction set was particularly reduced. I guess what qualified it as RISC was that the instructions were all one word and executed in one cycle. The Motorola 68020 was a CISC machine (Complex Instruction Set Computer). The instructions were variable length and executed in multiple cycles on variable length data. The IBM 360 was a CISC machine for the same reasons, although other computers like the UNIVAC 1108 worked according to the way I just described as RISC, since it had single word instructions that executed in one cycle, but didn't get that title.

The Sun SPARC architecture was designed with the C language in mind. C has a run-time stack for passing arguments to subroutines and for holding local variables. In general, CPUs are faster than memories, so instructions that operate on registers inside the CPU are faster than instructions that access memory. That's normal for most computers. The SPARC went one step further by having a very large number of registers and then mapping a local stack frame to a group of them. So, references to the stack could access registers instead of memory and go faster. When another subroutine is called, the new stack frame is mapped to another set of registers. They referred to that as a sliding window. This was effective because C programs generally have a lot of calls and returns with stack frames that are usually pretty small. Of course, there is memory behind the register stack and that's used when the registers run out. To switch context to another task, or in the case of TWOS, to another object, the registers all have to be saved, and you needed Assembler Language code to do that.

The Sun Workstation could do graphics on its big screen. I made a way for TimeWarp on the Mark III Hypercube to make a log of all the events on each node in its local memory and, at the end of a simulation, dump all the logs out into a file. Discrete Event Simulation works by scheduling events in the future. The simulation engine advances the simulation to the time of the next scheduled event and then runs the code for the event. That in turn can schedule more events, and so it goes. The idea of a TimeWarp distributed simulation is that each processing node can move ahead in simulation time independently of the other nodes. But then a node can receive a message for an event that's in the past relative to that node's current simulation time. When that happens, the future messages have to be canceled and the simulation rolled back. That's accomplished by sending anti-messages. When messages and anti-messages collide they're annihilated. This strategy was called optimistic because the future messages can be right.

My log entries contained both real time and simulation time, the sending and receiving nodes, and the type of message, either positive or negative. I wrote a program to do a two-dimensional display of that information, using the two times for x and y coordinates, and the two nodes for the ends of lines. When I ran that, it made a graphic of the whole simulation on one screen. You could see messages going forward and backward.

At that time, the Sun screen was monochrome. But I found a Silicon Graphics IRIS color graphics workstation in a room by itself with nobody using it. Again, JPL was so cool! There was a little room with an IRIS workstation and a chair just across the hall from my office. I took it over and made my program work on it, using colors for the lines, green for positive messages and red for negative messages.

That made the graphic simulation results much better. Dave Jefferson presented papers at simulation conferences and used images

that I produced in h
is why when one da
was showing them a
to me he said: "*Tha*
we always get someth

I created the Tir
Graph and the Time
then on the IRIS w

Steve Bellenot c
idea for a different v
a circle with positio
the 7-dimensional
would go from on
trigonometry, whic
IRIS workstation n
modified my graphi
color there. Instead
the TimeWarp mes
like a brain workin

We called it the
did something prac
in time where no m
The underlying so
We reported this p
they were able to r
problem with a pla

Steve wrote a p
Multiconference of
Tools For Measurin
Distributed Simulat
the California Insti
was a joint project
was the administra
were several other
from JPL. They we

that I produced in his presentations. I didn't get to go. But I think this is why when one day there were some VIP visitors, and the manager was showing them around, introducing everybody, and when he came to me he said: "*That's Mike. We just let him do whatever he wants and we always get something good.*"

I created the TimeWarp Message Graph, the TimeWarp Execution Graph and the TimeWarp Object Activity Display, first on the Sun, and then on the IRIS workstation.

Steve Bellenot came back for another Summer. This time he had an idea for a different way to display my simulation log data. He proposed a circle with positions around the circle that represented the nodes of the 7-dimensional Hypercube. Messages from one node to another would go from one place on the circle to another one. That takes trigonometry, which he not I could easily do. We always preferred the IRIS workstation now because of its color display, so we used that. We modified my graphics program to draw a circle and plot the messages in color there. Instead of a static graph, it made an animated 2D display of the TimeWarp message activity that was almost like a video. It looked like a brain working. Everyone wanted to see it.

We called it the *HyperCircle*. It was very impressive to see, but it also did something practical. As it was running, there appeared a small gap in time where no messages were sent or received, and it was very visible. The underlying software was developed by a group at the main Lab. We reported this phenomenon to them and, knowing what to look for, they were able to reproduce it, and thus able to fix it. We found a real problem with a play program.

Steve wrote a paper about this and presented it at the 1989 Eastern Multiconference of the Society for Computer Simulation. He called it *Tools For Measuring The Performance and Diagnosing The Behavior Of Distributed Simulations Using Time Warp*. He made me his co-author from the California Institute of Technology. It wasn't wrong. The HyperCube was a joint project between JPL and Caltech, and besides that, Caltech was the administrator of JPL and my paychecks came from Caltech. There were several other papers where I was credited as one of the co-authors from JPL. They were all written by others in our team. I only did software.