Ah, Dr. Fox, a brilliant researcher and also the strangest I ever saw or heard of. This once theoretical physicist has left few stones unturned in the field of high performance computing. But with all his brilliance, as a scrivener Fox was one of those beings of whom nothing is ascertainable.

Ere introducing the researcher, as he first appeared to me, it is fit I make some mention of myself because some such description is indispensable to an adequate understanding of the chief character and topic about to be presented.

The nature of my avocations over the years has driven me quite mad. For you see, being contained for long hours in non-environment controlled rooms while being irradiated by the very devices of my empowerment had the unfortunate effect of complete isolation, even when among infinite aisles of fellow bodies. A shattered mind was my savior as my singular voice was joined by an ensemble. I quickly found refuge by anthropomorphizing whatever was available to me. I have had many lively conversations with my keyboard, and my mouse on numerous occasions has threatened to sue me for sexual harassment.

I maintain that I am crazy, not stupid (though I have been informed by my stapler that there is a legitimate, on-going debate currently on the latter). I know far well that it is impossible for my coffee mug to speak to me and lecture me on lowering my caffeine intake, and that my mouse's claims of sexual harassment would never hold up in court. However, distinguishing reality becomes much more difficult in the case where the source of the voice was human to begin with. A voice from a book or email sounds as sweet as the real thing. I still consider myself fortunate that I can distinguish between different voices (say for instance the voice I use to write this paper from my stapler), but I have still lost my personal identity as I can no longer separate the voices of my invention (from say reading a paper) from those that may actually be spoken to me from a real person. Thus I cannot be certain if I may actually know a person, or only have experienced a person through an indirect medium.

I must question my own accounts with Fox. While the voice seems real, I cannot comprehend how I may have come to a personal association with him, unless by pure chance. So in the case of Fox, I feel compelled to give him the benefit of the doubt and try to believe I never met him. Thus I must assume I only know him through the indirect medium of writing. But in this writing, one could not distinguish that fact.

I came to association with Fox by direction of another voice. "Go forth and seek out High Performance Commodity Computing" resonated between every fissure and valley in my fragmented mind.
Every fiber in my being knew to obey, despite my coffee mug's attempt to convince me that it was yet another caffeine induced hallucination.

Dr. Fox on the surface seemed to be an eager man. At first, Fox did an extraordinary quantity of research. As if long famishing for something to eat, he seemed to gorge himself on any and every topic without pause for digestion. I might speak of his day and night runs, but alas in our vocation, our only source of light and warmth is the soft life glow of our gateway which we keep constant vigilance with, rendering the point moot. But despite any positive impressions, I was forced to deal with a difficult truth, he wrote on silently, palely, mechanically.

Perhaps now is a good time to start my story. In my casual search to find information about High Performance Commodity Computing, I soon discovered that it is a field that seems to be driven only by one person, and of course that person is Geoffrey Fox. However, this topic is simply one of countless others that he has found involvement in. But I would assume I could do no better than with the world's expert on this topic.

"So please tell me of High Performance Commodity Computing," I asked.

Fox: HPcc, not to be confused with HPCC (High Performance Computers and Communication), nor HPè (High Performance CORBA), nor anything else outside my field like HPCC (High Performance Cluster Computing), is an approach to use commodity off-the-shelf software technologies in the construction of computational grids. Consider a 3 tier architecture. At the first tier (top level), this model provides a customizable client tier, consisting of components such as graphical user interfaces, application programs, and collaboration tools.

The middle tier is often referred to as an application server. It consists of high-level services such as load-balancing, integration of legacy systems, translation services, metering, and monitoring. An important aspect of the middle tier is that it both defines interfaces and provides a control function, coordinating requests across one or more lower-tier servers.

The bottom tier provides back-end services, such as traditional relational and object databases. A set of standard interfaces allows a rich set of custom applications to be built with appropriate client and middleware
The traditional model is a client-server architecture, often portrayed by an HTTP server invoking CGI scripts and linking to databases or other services on the back-end.
However, with the infusion of distributed object architectures, we can create more sophisticated and powerful services.

Using existing popular distributed-object architectures, such as CORBA, DCOM, and JavaBeans, the resulting 3 tier architecture will prove to be the most appropriate way to implement a range of computational grid environments.

“But wait,” I thought. CORBA? JavaBeans? DCOM? What are these?

Now and then, in the haste of my business, it had been my habit to assume that when an distant topic is discussed, it is explained. In my haste and natural expectancy of instant compliance, I looked up with my normal dumbfounded face, and nervously expected for him to inform me of some of these aberrations he spoke of. But Fox moved forward with his business without the least delay.

Perhaps with all his brilliance, he is simply lethargic to the possibility that I or any other audience may be uninformed on these topics. So I called to him, rapidly stating what it was I wanted him to do. Imagine my surprise, nay, my consternation, when without moving from his privacy, Fox in a singularly mild, firm voice, replied, "I would prefer not to."

"Prefer not to," echoed I. "What do you mean? All I require is some background information."

"I would prefer not to," said he.
Did I miss something? Did he already speak of this and my limited faculties prevented me from noticing? Perhaps I offended him by failing to pick up on this. But no, I checked and double checked, and there was no mention of these topics.

Is it possible that he resents lecturing somebody so far removed and beneath him? I looked at him steadfastly. His face was leanly composed. Not a wrinkle of agitation rippled him. Had there been the least uneasiness, anger, impatience or impertinence in his manner; in other words, had there been any thing ordinarily human about him, I would have accepted this as the case. But as it was, this was very strange. What was I to do? With the hurry of my business, I knew I needed to conclude the matter quickly. For the moment, I was on my own.

**An Interview With a Stapler**

"CORBA...I know I've seen this before. But where...and what was it?"

My stapler touted, "Gee, how dense are you? You saw it last week. You remember that on-going debate your office supplies are having?"

"Yes?"

"I'm changing my vote."

"Well, please, enlighten me."

Stapler: CORBA stands for “Common Object Request Broker Architecture”. It is a distributed object system.

Me: “Okay, but what does that really mean, and why would I care?"

Stapler: Boy, you are dense. Look, there are two trends in computing. First, programmers are *gaga* over Object-Oriented Programming. Many programmers would see OOP as the best thing since sliced bread, assuming they have seen sliced bread, since the primary diet seems to be pizza. It’s not to say that OOP isn’t nice. It lends itself to all those black box metaphors and so forth. But that’s the first trend.

The second trend is networks. People want to use or get stuff from remote computers. Why waste your own resources, when you can waste somebody else’s?

Me: “You seem awfully cynical.”

Stapler: Get used to it. Now put the two ideas together, and you get a distributed object system. Imagine calling some object you want to use, but it is sitting somewhere on some distant computer. This distributed object system will let you call the
object and it will magically find what you’re looking for and magically let you use it as if it were on your own machine.

Stapler: Now if you try poking around in that Swiss cheese you call your brain, you might recall that CORBA is one particular incarnation of a distributed object system. It was designed by the Object Management Group, a consortium of about 800 member groups. One of its purposes was to create a universal standard everybody could use to prevent problems of fractioning (ahem, Microsoft).

At the heart of CORBA is the ORB (Object Request Broker). ORBS are responsible for transporting requests and data to the correct places and also handle interoperability between languages because CORBA is supposed to support multiple languages. This is handled by the IDL (interface definition language). You also have your stub which lets you call a remote object, and your skeleton which is the implementation on the remote side.

ORBs also need to communicate with one another, so to ensure interoperability, they have the General Inter-ORB Protocol (GIOP). This does not specify the transport mechanism for ORBs, so the Internet Inter-ORB Protocol (IIOP) is specified. It provides a transport layer for sending GIOP messages over TCP/IP.
Java Beans

“So, what’s going on with all this Java stuff?” I asked my verbally abusive stapler.

Stapler: You’re not very bright. Unlike CORBA, that wasn’t covered at all. Why don’t you check out some books on it? Not that I expect that any of the books will actually help you, but you’ve got nothing to lose.

Java Beans, I Am

Do you like Java Beans, I am?

No I do not like Java Beans, nor your verbal spam.
I do not like them in a house,
I do not like them with a mouse.
I do not like them here,
I do not like them there.
I do not like them anywhere.
I do not like Java Beans and spam.

You do not like them,
So you say.
Try them! Try them!
And you may.

When we talk component models,
It's as easy as pouring a bottle.

Just use the GUI,
and avoid a lot of traditional hooey.

Drag and drop your object Beans,
and make your friends envious green.

Take an object motor that can float,
and drag it onto a plain ordinary boat.
Lo and behold, you have a motorboat.

With lean tools like the BeanBox,
Beans are keen, says Dr. Fox.
They're even good for Goldielocks.

With a gaping mouth, I managed to inquire, "What's going on here?"

Stapler: Java Beans are allowing programmers to create programs by using a GUI interface simply by dragging and dropping Beans, which are specially defined Java objects. It looks similar to other programs like Visual Basic. The BeanBox is Sun's GUI tool that allows you to do this. There are other third party tools you can use in its place.

I do not like Java Beans and spam.
I need Java Beans like I need a bomb,
Especially since I already have Microsoft DCOM

But Java is more fun.
It installs on an Intel,
It runs on a Sun.
It will run here,
It will run there.
It will run anywhere.
But wait, but wait!
I'm not taking that bait.
Objects must be able to communicate.
You're naming something Java cannot do.
Shame! Shame on you!

What you say is true,
That is, until there was Java version two.

We now have reflection.
We now have introspection.
And of course we have customization.

Stapler: By the way, the marketing name for "version 2" of Java was Java 1.1. And yes, I can see from that usual blank stare you have, you need an explanation.

A Java Bean is a special form of an object. In Java 1.0, objects were not able to interact with one another dynamically as they try to do here by simply combining them in certain ways. Sun often gives the example, two Java applets cannot communicate with each other. With the next version of Java, they tried to solve this problem while trying not to change the Java language too much, so they introduced a special set of rules to make special objects that can communicate, called Beans. A Bean must publish its properties (for instance, visual characteristics like size or color), its methods (what functions it knows how to provide), and its events (actions, such as mouse clicks).

So, reflection is a simple way to do this. By following some naming conventions, Java can automatically find these properties.

```java
public <PropertyType> get<PropertyName>()
public void set<PropertyName>(<PropertyType> x)

public String getLabel () { }
public void setLabel (String newLabel) { }
```

If you do not wish to use the default naming convention, and want more overall control, a BeanInfo Interface is available for you to write your own BeanInfo class. This is called “Introspection”. 
But what if I want to save the current state and then restore it at a later date? This will cause me lots of trouble, Because I want this code finished on the double. And since Java must be fully portable, I have to be careful that my code is supportable.

There will be no double trouble on the portable supportable.

This supplement is realizable, Just use "implements Serializable".

Stapler: Beans have a way to store the current state and properties. This is called “persistence”. Java being nice lets you accomplish this with the Serializable interface.

public class SmallBean implements Serializable

This works well for primitive types, but for more complicated types (classes, references, arrays), you will have to do extra work to define how they are handled.

JavaBeans rely on the Java stream classes, the ObjectOutputStream and ObjectInputStream classes for saving and retrieving their state information. These files are given the extension .SER. These can also be embedded in a .JAR file which is like a .ZIP file. This approach is similar to ActiveX’s .CAB files.

One more note on Java Beans. To provide seamless integration into Java, they take advantage of the event model that was introduced in Java 1.1. The Java event model is comprised of event objects, event listeners, and event sources. There is nothing in the event model specific to beans so they behave just like anything else in Java.
Java RMI

"Hey, look. The section title just changed. So if that was Java Beans, why do we have another topic on Java?"

Stapler: You’re not paying attention. Look, remember in CORBA, I outlined the fact that there are two trends, not one. Java Beans is your object component for OOP. Java RMI (Remote Invocation Method) is the network part.

“So Java Beans and Java RMI are completely separate topics?”

Stapler: Yes and no. Since Java tries to make everything seamless and natural, Beans and RMI can be used together. But you don’t necessarily need Beans to use Java RMI. And you can use Beans locally in which case you do not need RMI.

I do not want to pout,
Nor do I wish to be a lout,
But what if I need to route?
On the idea of remote invocations,
How hard is it to deal with distant locations?
I do not want to program sockets.
I do not want to deal with traffic.
I do not want to program ports or IP packets

No need to fear,
Java RMI is here.
No need to mock-it or abort,
For you need not worry about the details of losing
    a packet in some pocket on a socket on a port,
So I report.

RMI works on a higher level,
I assure you I speak not on the bevel.

Your ports, aborts, and reports
and sockets and pockets
and packets and traffic
are bundled together in a magical package.

You'll love your stubs,
Skeletons you'll hug.

And for actual data to send,
Just call on your old friend.
For it's still realizable,
you just implement Serializable.

“Okay. So if I were to draw an analogy to CORBA, CORBA has skeletons and stubs, so Java RMI also has skeletons and stubs. However, CORBA also uses an IDL, but since Java is one language, it does not require an IDL?”

Stapler: Congratulations, you finally got something right. You are correct that Java RMI is different because it only needs to support one language (Java). As such, the requirement for an IDL is removed.

Java RMI like other systems tries to make remote invocation simple. They provide a java.rmi package which lets you hide/ignore all the nasty details of setting up sockets and so forth.

Similar to CORBA, there is the notion of stubs and skeletons. And in the simple case, objects can be easily exchanged (transported) by your old friend, the java.io.Serializable implement.
The complete Java RMI system can be thought of as a four layer model.

Layer 1 is the application layer. It is the actual implementation of the client and server applications. Here the high level calls are made to access and export remote objects.

Layer 2 is the stub/skeleton layer. This layer is what the application deals with directly. All calls to remote methods and dealing with parameters and return objects are done through these proxies.

Layer 3 is the remote reference layer which is responsible for dealing with semantics of remote invocations. It also manages the communication between the client/server and virtual machine(s), (e.g., threading, garbage collection, etc.) for remote objects.

Layer 4 is the transport layer. This layer is responsible for actually setting up connections and handling the transport of data from one machine to another.

By the way, Java RMI lacks the notion of the ORB as in CORBA. It appears that a server is run that contains a registry which allows applications to look up objects that are being exported for remote method invocation.

**Enterprise Java Beans**

“Wait, I’m confused. We’ve already seen two Java sections. Why is there a third?”
Stapler: Because I lied to you. Here’s the secret: Java Beans and Java RMI are not meant to compete with CORBA. Java Beans was originally intended for intraprocess use. Typically, it was targeted for manipulating components and widgets through GUIs, and are typically client side processes.

Enterprise Java Beans is a server-side component model designed to be Sun’s response to CORBA and DCOM. Though Sun would like to claim EJB is an extension of Java Beans, but they are drastically different, and EJB is significantly more complicated. But the basic understanding of Java Beans and Java RMI may assist in an intuitive understanding of EJB.

One thing EJB attempts to solve is the difficulty with large scale, high volume, mission critical environments with ORBS. One of the problems with ORBs is that they do not define an “operating system”. They are simply communications backbones that are used to access and interact with unique remote objects. When developing a distributed object application using an ORB, you (the programmer) are responsible for concurrency, transactions, resource management, and fault tolerance. These services may be available and implemented in an ORB, but the application developer is still responsible for incorporating them into the business objects. Thus, developing the infrastructure required to handle things like concurrency, transactions, security, persistence, etc., is a great ordeal most corporate development teams are not equipped to handle.

A much older system (30 years) called Transaction Processing monitors have had more success at large scale mission critical applications (CICS and TUXEDO). TP monitors are operating systems for business systems, typically whose applications are written in languages like COBOL. As an operating system, they control and automatically manage the entire environment that a system runs in, including transactions, resource management, and fault tolerance. These use RPC (Remote Procedure Calls) which is the non-OOP version of RMI, and thus are not object oriented. This means TP monitors are not as flexible, extensible, or reusable as a distributed object system. They usually consisted of dumb terminals (first tier), COBOL or PL/1 applications (middle), and some kind of database (DB2). On of the first TP monitors was created for Atlantic Power and Light. It created an online support environment to share concurrently applications services and information resources with the batch and time sharing operating systems environment. Today, TP monitor technology is still used in data management, network access, security systems, delivery order processing, airline reservations, and customer service. Use of TP monitor technology is considered to be a cost-effective alternative to upgrading database management systems or platform resources to provide this same functionality.

In 1999, Anne Thomas of the Patricia Seybold Group coined the term “component transaction monitor” (CTM). They proposed a hybrid of TP
monitors and ORBs. The intent is to make it easier for developers to create, use, and deploy business systems. CTMs provide an infrastructure that can automatically manage transactions, object distribution, concurrency, security, persistence, and resource management. They should be good for mission-critical work, but also be valuable to smaller systems because they are easy to use.

And so the CTM has found its way into EJB and has become the centerpiece to the technology. Sun suggests that ORBs create a vendor “lock in” which makes it really hard to swap out an ORB for a better one. One of Sun’s goals to allow EJB CTMs to be swapped and upgraded more easily.

My thoughts drifted for a moment. It seems odd that so much trouble was made to introduce Anne Thomas and this Patricia Seybold Group. The group seems to be some consultant business. I wonder if some authors are getting incentives to advertise for them.

Then it came to my attention. On my stapler’s back, it branded a faded logo: Sun Microsystems. Yes, I remember long ago, I once liberated a stapler from there. Can I trust this stapler’s statements? I choose to believe so. Odd though, how a stapler has come into my confidence when my vocation is virtual and paperless.

Refocusing my attention, I realized that Fox made no mention of EJB. And it appears that Fox’s solutions involve using Java with CORBA. I suppose, EJB would get speed advantages with its Java RMI model and integrated event driven models. Perhaps this will be something Fox follows up, given his voracious appetite.

So how do you like Java Beans?

I do so like Java Beans and spam.
Thank you!
Thank you, Sam-I-am.

Microsoft DCOM

"Should I move on to DCOM?" I asked.

"Not unless you're up to reading The Gates Who Stole Distributed Objects."

Pondering the decision for a moment, I decided against such a venture mainly due to time constraints.

Stapler: Here’s the quick run down. DCOM is Microsoft’s answer to CORBA. It’s bigger and much more complex than Java Beans and Java RMI (not EJB), but it does more. It lets you share both components and documents. They publish component functionality in .TLB files, but also requires use of your Windows
system registry, which means that DCOM is platform-dependent. However, DCOM is language independent, so they have their own IDLs, like CORBA.

The basic break down is, DCOM runs the fastest, Java Beans/RMI is the most elegant, and CORBA is the most complete solution. And everybody has found ways to get each to interoperate with itself.

Going back to Fox’s 3 tier model, since CORBA seems to be the best overall solution, look at CORBA as the middle tier. Then Java Beans will work with CORBA on the first tier, probably in an Applet through a web browser.

**Sleeping in the Light**

After thinking about what was left in HPcc, I realized that now with the structure, all that remained were the questions of how to improve performance within the structure.
I decided it was time to return to Dr. Fox. I spoke, “I think I understand the idea now. So would you tell me about improving performance in HPcc?”

But once again, a simple but blunt, “I would prefer not to,” rang in my ears.

“Why do you refuse?”

“I would prefer not to.”

Still, there was no change in the inclination of his voice. There was not a flinch of any mannerism hinting at any kind of malice.

I do not understand this behavior. I looked to my stapler and received the oddest image of it shrugging also in disbelief.

Needing to press on, I opened another dialogue with my stapler.

Me: Given that CORBA will probably be used in the middle tier, and that CORBA is not necessarily fast, optimization can be applied here to get better speed.

Stapler: Yes, that is true. CORBA is defined in terms of a set of facilities, where each facility defines an established standardized high level service. Facilities are split into those that are required by most applications (horizontal facilities) and those that are defined to promote interoperability within specific application domains (vertical facilities). It is expected that as CORBA evolves, some vertical facilities migrate into horizontal facilities. The idea is to introduce a new vertical facility that defines how CORBA objects should interact with one another in a high performance environment. Since CORBA currently supports relatively simple computing models including “embarrassingly parallel” activities of transaction processing or data flow, an HPcc facility should fill the gap.
Me: Is it successful?

Stapler: It doesn’t look like it has been done yet.

Me: And we can speed up the back-end layer too, correct? We would want to replace those commodity services by a high performance version if it poses a bottleneck in performance.

Stapler: Yes, you would replace sequential databases with parallel database machines, and sequential or socket-based messaging is replaced with message-passing implementations on low-latency, high-bandwidth dedicated parallel machines.
Me: Is there anything left to optimize?

Stapler: Of course, there are a thousand things you can do. However, Java seems to be an obsession of Fox’s and it’s true that Java has a lot of potential to be optimized.

Java has the potential to become an important language in high performance computing. Many programmers know C++ and can migrate to Java. Many new students learn Java. Furthermore, alternative languages for high performance computing are not as captivating. New programmers will probably not learn Fortran, while C++ is difficult to get good performance with parallel code.

Performance is an issue with Java, particularly because of the virtual machine constraint. However, advancements in JIT compilers and native-class libraries can improve performance. There is also the possibility of developing full Java compilers.

The exception handlers in Java risk performance, but the lack of pointers make optimization much easier.

Looking at four forms of parallelism shows that with some work, Java can support all of these.

Data Parallelism, meaning natural large scale parallelism found from parallel updates of grid-points, particles, and other basic components in scientific computations might be exploited. Java doesn’t naturally contain this, but it is believed it could be extended like HPF or HPC++.
Modest Grain Size functional parallelism refers to when unique application functions can be executed concurrently, like a web browser doing I/O while doing computation. Java supports threads, so this is possible.

Object Parallelism is natural for Java. Java can use the applet mechanism to portably represent objects.

Metaproblem parallelism occurs in applications that are made up of several different sub problems which themselves may be sequential or parallel. There is nothing specific in Java preventing any optimizations in this area.

So it is believed a High Performance version of Java is realizable. Getting more performance may require a true compiler rather than byte code, but there is no technological reason this cannot be done.

Currently, the first components of HPJava have been released. mpiJava is the first piece of this puzzle to be released. It includes extensible framework for SPMD programming. Language level support for things like distributed arrays are being added because without it, the full potential of the library cannot be exploited. mpiJava itself does not assume any special extensions to the Java language; it should be portable to any platform that provides compatible Java-development and native MPI environments.

Currently mpiJava 1.2 provides the full functionality of MPI 1.1. It is implemented as a set of JNI wrappers to native MPI packages. Platforms currently supported include Solaris using MPICH or SunHPC-MPI, Linux using MPICH, and Windows NT using WMPI 1.1. The new release enables communication of object types by object serialization.

There are also other versions of Java MPI being developed. None have yet achieved full compliance with the draft specification, though mpiJava seems to be closest to being the first.

Sun Burnt

Webflow

Fox has made references to applications called Webflow and TANGO. Perhaps these are working examples of this very topic. By the mere mention of them offers insights into his soul. Maybe tucked away, either buried in shadows of ego or apathy, lay a noble spirit willing to help. Perhaps I can free it.

Dr. Fox, I see you make mention to Webflow. I am curious about it.

Fox never turned his head, frowned his eyes, nor released a sigh. He simply continued with his own business.
I tried again, “Dr. Fox, Webflow is an interesting subject. What are your thoughts about it in relation to HPcc?”

Again, there was silence. I began to see that I would not get a response to an open-ended question. But with the other kind, I already knew the response.

Dr. Fox, would you please explain Webflow?

“I would prefer not too.”, spoke his mild cadaverous self.

Me: I thought as much. Okay Stapler, what’s the door prize this time?

Stapler: You are the proud winner of yet another buzz word, *Pragmatic Object Web*. This time the term HPcc has dropped off the face of the earth and HPCC has resurrected from the dead. This new term seems to describe building distributed systems from, yes, again, CORBA, DCOM, and Java Beans. But we now how the additional connotation that these technologies are leading to merged standards.

Me: “So this is an example of HPcc; the three tier model, and all the bells and whistles?”

Stapler: That may be a reasonable answer.

Me: “Does this program do anything?”

Stapler: Good question. It seems they are interested in using a GUI interface to allow users to easily create things like physics applications. The first version consists of a suite call Quantum Simulations of Condensed Matter Systems mostly comprised of Monte Carlo programs. A later incarnation is boasting a GUI image filtering program.
Stapler: The first GUI interface may have been a custom version. Later versions may have been commercial implementations. Fox lists VisualStudio from JavaSoft, VisualAge from IBM, VisualModeler from Microsoft, VisualCafe from Symantic, and Rational Rose from Rational. Fox seems to be waiting for a new standard to replace these called the Uniform Modeling Language (UML), led by Rational, supported by Microsoft, and adopted by the OMG for CORBA.

Me: Strange how a web search points VisualStudio and VisualModeler both to Microsoft. But IBM has information. Yes, it seems to be a glorified BeanBox, with lots of database connectivity features. And it looks like EJB made it into these tools after all.

Stapler: The GUI part specifies the front end for the 3 tier model. For the Middleware, Fox has created his own ORB written in Java using a JavaIDL, called JWORB (Java Web Object Request Broker). Though there are other Java ORBs available, he seems to squeeze more performance out of his. It’s supposed to act both as a Java Web Server and as an ORB for Java objects as well as act as a CORBA client or server for Java objects.

Okay, so what’s a Java Web Server? And is it only his ORB that has this dual functionality, or is it all of them?

Stapler: Yet another distant topic. However, the main idea is they use things called Servlets which are server-side components and they are supposed to replace CGI scripts because CGI is not dynamic. And for the second question, I don’t know.

Well, I would hope it has some kind of additional functionality seeing that he has a C++ ORB (omniORB2) which outperforms the Java stuff by a factor of 20.

Stapler: Well that leaves the last tier, the Back End, and it seems pretty usual by Fox’s standards.
TANGO

Me: So onto Tango. Please tell me what’s in store.

Stapler: Sorry, I can’t. It’s server error 404.

The few pieces I was able to find seem to old to offer any real insights into HPcc. The project seems to predate Java Beans which doesn’t seem useful given Fox’s current direction with the topic. Given that, I decided to move on.

Ah humanity!

My time came to an end, and I was expected to share what I’ve learned. Unfortunately, I still do not understand. There are so many questions to be asked and holes in my comprehension.

As difficult as Fox is, I cannot say he was pretentious or malicious. He was never angry, or smug, or bitter, or sad. He never made any kind of judgment at all. Fox is
absolutely pure. And yet I find that unsettling. By his innocence, my life is a mess and he
resigns my life. And now, I must accept my day and death as a lollipop.

Some links I actually remembered to write down:

High Performance Commodity Computing
http://www.npac.syr.edu/users/gcf/HPcc/HPcc.html
http://www.npac.syr.edu/users/haupt/SC97/Hpccdemos.html
http://www.npac.syr.edu/users/gcf/

HPCC
http://renoir.csc.ncsu.edu/MRA/HTML/Workshop2/Fox2/index.html

Java Beans/RMI, DCOM, CORBA
http://www.zdnet.com/pcmag/features/beans/_open.html
http://www.zdnet.com/devhead/stories/articles/0,4413,2144454,00.html
http://www.javacoffeebreak.com/articles/javarmi/javarmi.html
http://www.zdnet.com/devhead/stories/articles/0,4413,1600418,00.html
http://www.webadvisor.com/activex.html
http://java.sun.com

Jeff’s Wonderful CORBA Talk!
http://www.cse.ucsd.edu/classes/sp00/cse225/notes/jeff/talk.html

Transaction Processing Monitors
http://www.sei.cmu.edu/str/descriptions/tpmt.html
Bea Systems
http://www.beasys.com/products/tuxedo/

HPJava
http://www.npac.syr.edu/projects/pcrc/HPJava/

mpiJava
http://www.npac.syr.edu/projects/pcrc/mpiJava/mpiJava.html

From Javabean to Database Web Servers in the Pragmatic Object Web
http://www.npac.syr.edu/users/gcf/working/wiscpaper/wisc98.html

WebFlow - High-Level Programming Environment and Visual Authoring Toolkit for
High Performance Distributed Computing
http://www.supercomp.org/sc98/TechPapers/sc98_FullAbstracts/Akarsu809/Index.htm
Introduction To Java Web Object Request Broker (JWORB)
http://www.npac.syr.edu/users/timucin/misc/jworb/

Quantum Simulations of Condensed Matter Systems
http://www.ncsa.uiuc.edu/Apps/CMP/cmp-homepage.html

Some Books


Enterprise Java Beans, Richard Monson-Haefel, 2000

Java RMI, Troy Bryan Downing, IDG Books, 1998

The Grid, Morgan Kaufmann Publishers, Inc. (Ch.10)

My additional apologies to:
Herman Melville (Bartleby)
Theodor Seuss Geisel (Green Eggs and Ham)
The real Dr. Geoffrey C. Fox, whom I never met. By no means do I actually believe there is any resemblance to my fictitious Bartleby metaphor.