A Dialogue on I/O Scheduling in Xen

[The following is a word-for-word transcription of what happened in class today.]

Socrates: One, two, three... Yes, everyone appears to be here, and everything is in order. But I am feeling very ill. Timaeus, however, knows all about this topic. Timaeus, would you care to lead the discussion?

Timaeus: Well, I'm quite taken aback, as I was planning on just being more or less passive today. But I am also greatly honored, and I would be remiss to decline considering all the hospitality you have shown us.

Now, let us remind ourselves how we concluded last time. I believe that we had come to a sure conclusion on the nature of the just State and were just about to discuss how to properly schedule I/O in Xen. There was an interesting paper published not to long ago on this topic. Is everyone familiar with it?

Thrasymachus: Yes, yes. I'm very glad you brought that paper up, because I didn't really understand the relationship between Xen's credit scheduler, lottery scheduling, and stride scheduling.

Timaeus: Ah, yes. Can anyone explain how the credit scheduler works?

Glaucon: The paper does an adequate job of this, but yes, surely it is good to review. Essentially, each domain receives a number of credits. Every 10ms quantum, Xen deducts 100 credits from the running domain. Domains are either UNDER their credit limit or OVER; when Xen picks the next domain to run, it picks one of the UNDER domains. Once the sum of all domains' credit levels reaches a negative number, everyone is reset to their usual number of credits. Did I summarize it correctly?

Timaeus: You did.

Glaucon: Excellent! I'm good at this!

Timaeus: What about stride scheduling? Can anyone explain stride scheduling? I see only clueless looks.

Socrates: <explains stride scheduling>

Timaeus: Ah, Socrates, I see that illness has not dulled your wit. Now, why do we need the BOOST state?

Polemarchus: BOOST is a state added to UNDER and OVER that allows a domain with a pending virtual interrupt to receive priority over domains in UNDER. So, when enqueuing a virtual interrupt, Xen can set the target domain to BOOST. This helps decrease latency for IO-bound applications.

Timaeus: Excellent exposition, my dear Polemarchus.

Glaucon: I'm confused. When does the domain go out of BOOST state, exactly?

Timaeus: No one appears to know. It is a mystery. However, one thing is certain: you don't always get BOOSTed. Only if you are already UNDER can you receive BOOST, not if you are OVER. Now, from this, a number of things can be said. Chief among them is the fact that it seems like a bad idea to run CPU-intensive and IO-intensive tasks in the same domain, since the former will use up all the credits and prevent the domain from becoming BOOSTed. Don't you agree, Socrates.

Socrates: Timaeus, you have put the matter in a most excellent way. Perhaps, if the scheduler were smarter, it could predict whether a domain is running latency-sensitive or CPU-bound tasks, and act accordingly.

Timaeus: An excellent point, dear Socrates. So, does everyone more or less understand how the Xen scheduler works? Good. Let's move on to event channels. Now, can someone explain how the hypervisor communicates events to a guest?

Thrasymachus: It puts it into an event queue in shared memory and notifies the domain over an event channel.

Timaeus: And what was the issue with this brought up in the paper?

Thrasymachus: Well, you see, the event ports are listed in order of boot – that is, in the order in which the domains boot – and Xen originally searched this list in order, so domains that booted first were given preference. The authors of this paper contributed a patch so that Xen remembers where it left off searching the list and resumes from that point. They are very proud of this fact and even give the changeset number.

Timaeus: Yes, yes, changeset 324. What else could they have done?

Polemarchus: I suppose they could have used some randomness or done FIFO with events.

Timaeus: And when we talk about “tickling” a scheduler, what does that mean exactly?

Glaucon: I believe it means that you yield to the scheduler, telling it to pick someone else to run. This is done when an event arrives, to give the receiving domain a chance to run sooner.

Timaeus: Precisely! Why does this cause problems?

Glaucon: Priority inversion could occur as described in Section 4.2. Also, the latency of a higher priority interrupt could increase.

Timaeus: Hmm, indeed.

Thrasymachus: I just thought of something. What proportion of credits do you suppose should be given to guests and driver domains? You can't give dom0 maximum priority because an IO-intensive domain would keep scheduling dom0; dom0 would run all the time and starve everyone else.

Timaeus: I don't think this issue is explored in the paper.

Thrasymachus: And what if dom0 is OVER and the other domains are UNDER? Everyone is blocked for IO and dom0 can't be run.

Timaeus: There are two points at which you get new credits. First, when everybody is OVER, and second, every so often.

Thrasymachus: Oh.

Glaucon: Can you get scheduled if you are OVER?

Timaeus: Yes, if everyone is OVER.

Socrates: Well, we are out of time. Next time, we can discuss the nature of knowledge.