Topics: Time Warp

Time Warp is another way of parallelising programs, mostly applied in the area of discrete event simulation, using distributed architectures

Topics: Time Warp

Time Warp is another way of parallelising programs, mostly applied in the area of discrete event simulation, using distributed architectures

For example, simulating the movement of molecules in a fluid; or packets in a network; or tanks on a battlefield

Topics: Time Warp

Time Warp is another way of parallelising programs, mostly applied in the area of discrete event simulation, using distributed architectures

For example, simulating the movement of molecules in a fluid; or packets in a network; or tanks on a battlefield

"Discrete Event" means we simulate the system for discrete ticks of time, $t = 0$, $t = 1$, etc., where each tick might be a second or a microsecond; even a nano- or picosecond for some physics and chemistry simulations

Topics: Time Warp

Time Warp is another way of parallelising programs, mostly applied in the area of discrete event simulation, using distributed architectures

For example, simulating the movement of molecules in a fluid; or packets in a network; or tanks on a battlefield

"Discrete Event" means we simulate the system for discrete ticks of time, $t = 0$, $t = 1$, etc., where each tick might be a second or a microsecond; even a nano- or picosecond for some physics and chemistry simulations

The objects in the system interact via events, e.g., one molecule hitting another or a packet entering a router

Time Warp

On the face of it, simulation should be good to parallelise, as it comprises objects (molecules, packets, tanks) that are mostly independent, but require the occasional interaction (molecules bounce off each other; packets overload a router; tanks fire missiles at each other)

Time Warp

On the face of it, simulation should be good to parallelise, as it comprises objects (molecules, packets, tanks) that are mostly independent, but require the occasional interaction (molecules bounce off each other; packets overload a router; tanks fire missiles at each other)

As long as there isn't *too* much interaction (lots of missiles?)

Time Warp

Simulations can have millions of objects, and so must be done on a distributed memory machine: they simply won't fit on a shared memory machine

Time Warp

Simulations can have millions of objects, and so must be done on a distributed memory machine: they simply won't fit on a shared memory machine

This is parallelism for reasons of size, not speed

Time Warp

Simulations can have millions of objects, and so must be done on a distributed memory machine: they simply won't fit on a shared memory machine

This is parallelism for reasons of size, not speed

So the events must be messages

Time Warp

Simulations can have millions of objects, and so must be done on a distributed memory machine: they simply won't fit on a shared memory machine

This is parallelism for reasons of size, not speed

So the events must be messages

Now the problem turns out to be synchronisation, but in *simulation* time

Time Warp

Two molecules A and B might be on separate processors; each moving through their simulated ticks of time

Time Warp

Two molecules A and B might be on separate processors; each moving through their simulated ticks of time

Molecule A might be trundling along, simulated at time $t = 100$, $t = 101$, etc., only to find it should have bounced off molecule B at time $t = 90$

Time Warp

Two molecules A and B might be on separate processors; each moving through their simulated ticks of time

Molecule A might be trundling along, simulated at time $t = 100$, $t = 101$, etc., only to find it should have bounced off molecule B at time $t = 90$

The “collision” message may simply have taken ages to get from B to A across the cluster’s network

Time Warp

Two molecules A and B might be on separate processors; each moving through their simulated ticks of time

Molecule A might be trundling along, simulated at time $t = 100$, $t = 101$, etc., only to find it should have bounced off molecule B at time $t = 90$

The “collision” message may simply have taken ages to get from B to A across the cluster’s network

Or B’s processor is slower than A’s; or more heavily loaded; etc.

Time Warp

So, we could just synchronise at every time step?

Time Warp

So, we could just synchronise at every time step?

Yes, but this would be very slow: just think of the synchronisation messaging needed

Time Warp

So, we could just synchronise at every time step?

Yes, but this would be very slow: just think of the synchronisation messaging needed

The secret is to let each processor work at its own pace and only synchronise when necessary

Time Warp

So, we could just synchronise at every time step?

Yes, but this would be very slow: just think of the synchronisation messaging needed

The secret is to let each processor work at its own pace and only synchronise when necessary

There are several ways of doing this, but the weirdest is the *Time Warp*

Time Warp

So, we could just synchronise at every time step?

Yes, but this would be very slow: just think of the synchronisation messaging needed

The secret is to let each processor work at its own pace and only synchronise when necessary

There are several ways of doing this, but the weirdest is the *Time Warp*

This is an *optimistic* method

Time Warp

Invented by Jefferson in the 1980s, it tries to extract as much parallelism as possible by each object optimistically simulating forward in time, only going back to fix things if they went too far

Time Warp

Invented by Jefferson in the 1980s, it tries to extract as much parallelism as possible by each object optimistically simulating forward in time, only going back to fix things if they went too far

For the most part, most objects do not interact, so waiting (synchronising) on other objects is usually a waste of time

Time Warp

Invented by Jefferson in the 1980s, it tries to extract as much parallelism as possible by each object optimistically simulating forward in time, only going back to fix things if they went too far

For the most part, most objects do not interact, so waiting (synchronising) on other objects is usually a waste of time

Every processor simulates at its own speed, under the optimistic assumption that there would have been no interaction or synchronising

Time Warp

So molecule A is simulated as fast as its processor permits, its time progressing $t = 100$, $t = 101$, etc.

Time Warp

So molecule A is simulated as fast as its processor permits, its time progressing $t = 100$, $t = 101$, etc.

Similarly for B on its processor $t = 89$, $t = 90$, etc.

Time Warp

So molecule A is simulated as fast as its processor permits, its time progressing $t = 100$, $t = 101$, etc.

Similarly for B on its processor $t = 89$, $t = 90$, etc.

Each molecule lives in its own bubble of time, independent of other molecules

Time Warp

So molecule A is simulated as fast as its processor permits, its time progressing $t = 100$, $t = 101$, etc.

Similarly for B on its processor $t = 89$, $t = 90$, etc.

Each molecule lives in its own bubble of time, independent of other molecules

Each object in the system has an *input queue* of messages yet to be processed from other objects: these messages will have timestamps in the future of the object

Time Warp

So molecule A is simulated as fast as its processor permits, its time progressing $t = 100$, $t = 101$, etc.

Similarly for B on its processor $t = 89$, $t = 90$, etc.

Each molecule lives in its own bubble of time, independent of other molecules

Each object in the system has an *input queue* of messages yet to be processed from other objects: these messages will have timestamps in the future of the object

An object will repeatedly read the next message from its queue and act upon it. The object's current time is set to the timestamp on the message

Time Warp

Now sometimes, and we hope this is rare, there *is* interaction and we might have gone too far: molecule A is at time 102 but then receives a message with timestamp $t = 98$, so A should have bounced then

Time Warp

Now sometimes, and we hope this is rare, there *is* interaction and we might have gone too far: molecule A is at time 102 but then receives a message with timestamp $t = 98$, so A should have bounced then

The bounce message simply didn't reach A soon enough

Time Warp

Now sometimes, and we hope this is rare, there *is* interaction and we might have gone too far: molecule A is at time 102 but then receives a message with timestamp $t = 98$, so A should have bounced then

The bounce message simply didn't reach A soon enough

In this case the Time Warp mechanism says we should locally reverse time and roll A back to time 98, do the computation for the bounce, then continue forward

Time Warp

To do this *rollback* we either need reversible computations, or we keep a record of the past states of A and revert it to its state at time 98

Time Warp

To do this *rollback* we either need reversible computations, or we keep a record of the past states of A and revert it to its state at time 98

A then can continue forward, after bouncing appropriately

Time Warp

To do this *rollback* we either need reversible computations, or we keep a record of the past states of A and revert it to its state at time 98

A then can continue forward, after bouncing appropriately

If rollbacks do not happen too often, we get improved speedup as each processor can compute at full speed with no synchronisation

Time Warp

But there is a problem

Time Warp

But there is a problem

Suppose A is optimistically computing forward and decides it hits molecule C at time $t = 101$

Time Warp

But there is a problem

Suppose A is optimistically computing forward and decides it hits molecule C at time $t = 101$

So it sends a message to C, saying “A hits C at $t = 101$”

Time Warp

But there is a problem

Suppose A is optimistically computing forward and decides it hits molecule C at time $t = 101$

So it sends a message to C, saying “A hits C at $t = 101$”

Then B’s message arrives, saying “B hits A at $t = 98$”

Time Warp

But there is a problem

Suppose A is optimistically computing forward and decides it hits molecule C at time $t = 101$

So it sends a message to C, saying "A hits C at $t = 101$"

Then B's message arrives, saying "B hits A at $t = 98$"

This bounce will probably divert A so that it would not have hit C

Time Warp

But there is a problem

Suppose A is optimistically computing forward and decides it hits molecule C at time $t = 101$

So it sends a message to C, saying "A hits C at $t = 101$"

Then B's message arrives, saying "B hits A at $t = 98$"

This bounce will probably divert A so that it would not have hit C

But the message has already been sent!

Time Warp

A’s message to C is now an error

Time Warp

A's message to C is now an error

Time Warp fixes this with *anti-messages*

Time Warp

A's message to C is now an error

Time Warp fixes this with *anti-messages*

As part of the rollback process, if an object finds it sent a message in error it now sends a corresponding anti-message

Time Warp

A's message to C is now an error

Time Warp fixes this with *anti-messages*

As part of the rollback process, if an object finds it sent a message in error it now sends a corresponding anti-message

A sends the anti-"A hits C at $t = 101$" message to C

Time Warp

Two things can happen in C:

Time Warp

Two things can happen in C:

- If the message arrives in C's future (C is still processing at time $t = 100$, say), the positive message is still waiting for C to read it. Thus the positive message can simply be removed from C's input queue of messages

Time Warp

Two things can happen in C:

- If the message arrives in C's future (C is still processing at time $t = 100$, say), the positive message is still waiting for C to read it. Thus the positive message can simply be removed from C's input queue of messages
- If the message arrives in C's present or past (C is processing at time $t = 104$, say), this triggers a rollback of C: it unwinds to time $t = 101$, and then proceeds forward, this time without the bounce off A

Time Warp

C's rollback might require anti-messages from C; which might trigger more rollbacks from other objects; and so on

Time Warp

C's rollback might require anti-messages from C; which might trigger more rollbacks from other objects; and so on

In the worst case, there can be an *anti-message cascade* where more and more objects trigger rollbacks of other objects

Time Warp

C's rollback might require anti-messages from C; which might trigger more rollbacks from other objects; and so on

In the worst case, there can be an *anti-message cascade* where more and more objects trigger rollbacks of other objects

The hope is that this is rare, and outweighed by the general forward progress from the optimistic computation

Time Warp

For the right kind of simulation, Time Warp is very effective

Time Warp

For the right kind of simulation, Time Warp is very effective

However, there are so many important details of implementation that it is hard to find a good implementation of Time Warp

Time Warp

For the right kind of simulation, Time Warp is very effective

However, there are so many important details of implementation that it is hard to find a good implementation of Time Warp

For example, the management of past states. These are required for the rollbacks, but you can't keep them forever as they will eat up more and more memory as the computation proceeds

Time Warp

For the right kind of simulation, Time Warp is very effective

However, there are so many important details of implementation that it is hard to find a good implementation of Time Warp

For example, the management of past states. These are required for the rollbacks, but you can't keep them forever as they will eat up more and more memory as the computation proceeds

So implementations include a garbage collection of old, inaccessible states

Time Warp

If all objects *and all unread messages* are beyond time $t = 100$, then there cannot be a rollback to earlier times

Time Warp

If all objects *and all unread messages* are beyond time $t = 100$, then there cannot be a rollback to earlier times

So states older than $t = 100$ can then be discarded (garbage collected) across the entire system

Time Warp

If all objects *and all unread messages* are beyond time $t = 100$, then there cannot be a rollback to earlier times

So states older than $t = 100$ can then be discarded (garbage collected) across the entire system

So now we have the problem of finding the earliest timestamp in the system: another problem in its own right as this is information that is distributed across the entire system

Time Warp

If all objects *and all unread messages* are beyond time $t = 100$, then there cannot be a rollback to earlier times

So states older than $t = 100$ can then be discarded (garbage collected) across the entire system

So now we have the problem of finding the earliest timestamp in the system: another problem in its own right as this is information that is distributed across the entire system

We have to be careful that the messaging needed to find the earliest timestamp plus the time spent garbage collecting is not too large in itself

Time Warp

Time Warp is effective if

- interactions are rare
- the cost of rollback is low

Time Warp

Time Warp is effective if

- interactions are rare
- the cost of rollback is low

But

- the cost of storing state can be high
- the overheads of garbage collection can be high

Time Warp

Time Warp is effective if

- interactions are rare
- the cost of rollback is low

But

- the cost of storing state can be high
- the overheads of garbage collection can be high

Other distributed simulation methods are more popular, e.g., conservative simulation: you only progress as far as you can prove is safe, which may be more appropriate than Time Warp where there is a moderate amount of interaction

Time Warp

There is a big literature on parallel discrete event simulation (PDES) as it is used by various large organisation, e.g., the US Army

Time Warp

There is a big literature on parallel discrete event simulation (PDES) as it is used by various large organisation, e.g., the US Army

Much research into Time Warp was funded by the US Army

Time Warp

There is a big literature on parallel discrete event simulation (PDES) as it is used by various large organisation, e.g., the US Army

Much research into Time Warp was funded by the US Army

They have very large battlefield simulations

Time Warp

There is a big literature on parallel discrete event simulation (PDES) as it is used by various large organisation, e.g., the US Army

Much research into Time Warp was funded by the US Army

They have very large battlefield simulations

Researchers were worried when explaining Time Warp to the Generals that, by talking about missiles and anti-missiles instead of messages and anti-messages that the Generals might get the wrong idea and require the invention of anti-missiles. . .