Temporal\_Error\_Caption

At time E1, a local process **p** on device **Bπ\_1** invokes the virtual time function **freeze** causing a write to the **/sys/vt/** file system triggering the **mode\_store** callback in the VT kernel module. In the callback, the **gpio\_direction\_output** function is called. At this time, **E2, Bπ\_1** emits a HIGH signal to the GPIO bus which triggers each devices’ **vtgpio\_irq\_handler\_freeze** as an interrupt inside the VT kernel module. The time between receiving the interrupt and freezing(pausing) the registered virtual time processes (and their clocks) is **E3 - E2** and is the overhead of freezing the distributed system. Technically, **E2 is** experienced uniformly while **E3**  will be experienced by different devices at various times depending on the number of processes that are in virtual time. This is due to having to send a **SIGSTOP** signal as well as writing to each processes virtual time fields in the **procfs** (process file system) later used to calculate clock offsets. Therefore more processes to freeze results in larger overhead, however this overhead does not translate directly to error as explained when processes resume. At some point **E3,** all processes are frozen and wait in this state for external off-line calculations to be performed.

When the parent process **p’** of originating process **p** on device **Bπ\_1** finishes the off-line calculation or receives the result from an external source, it writes to the **/sys/vt/** file system at time **E4** triggering the **mode\_store** callback in the VT kernel module. In the callback, the **gpio\_direction\_input** function is called returning the pin to a LOW state. At **E5** when the LOW signal is emitted all the devices connected to the GPIO bus receive a interrupt triggering each devices’ **vtgpio\_irq\_handler\_resume** interrupt inside the VT kernel module. Similarly to the time between **E2** and **E3**, the time between **E5** and **E6** is spent returning the processes to a running state and writing the total offset to each virtual time enabled process field. Each process will be running and paused for the same amount of time but are not guaranteed to have the same clocks between **E2** and **E3** and **E5** and **E6** respectively.

Overhead is incurred on the process invoking freeze between **E1** and **E2** and **E5** and **E6** as all other processes will be running while this process is in a blocking state. This is the overhead of invoking an offline calculation. The offline calculation time will be included in the total running time of the experiment but is transparent to any process registered in virtual time.

**Pausing Algorithm**

1. Off-line request created/triggered **E1**
2. Device **p** writes **“freeze” > /sys/VT/mode E1 → E2**
   1. sys\_write()
      1. fs stuff
      2. mode\_store()
         1. gpio\_direction\_output()
3. Hardware interrupt triggered on rising edge of 5v signal (sync barrier)  **E2 → E3**
   1. Pause\_pids
      1. for ( #pids ) {
         1. SIGSTOP $PID
         2. fs stuff
4. In frozen state / offline calculations  **E3 → E4**
5. Device **p** writes **“resume” > /sys/VT/mode E4 → E5**
   1. sys\_write()
      1. fs stuff
      2. mode\_store()
         1. gpio\_direction\_output()
         2. Gpio\_direction input()
6. Hardware interrupt triggered on rising edge of 5v signal (sync barrier)  **E5 → E6**
   1. Resume\_pids
      1. for ( #pids ) {
         1. fs stuff
         2. SIGCONT $PID
7. All procs continue  **E6 →**

**GTOD --gettimeofday() ftrace**

|  |  |
| --- | --- |
| In Virtual Time | NOT In Virtual Time |
| 0) | sys\_gettimeofday() {  0) | do\_gettimeofday() {  0) | getnstimeofday() {  0) 0.250 us | arch\_counter\_read();  0) | ns\_to\_timespec() {  0) 0.541 us | ns\_to\_timespec.part.0();  0) 3.041 us | }  0) 8.250 us | }  0) + 10.583 us | }  0) + 13.292 us | } | 1) | sys\_gettimeofday() {  1) | do\_gettimeofday() {  1) | getnstimeofday() {  1) 0.250 us | arch\_counter\_read();  1) 2.792 us | }  1) 5.333 us | }  1) 8.000 us | } |
| 17.387 us/GTOD (from user space)  13.333 us/SYS\_GTOD (kernel space) | 11.844 us/GTOD (from user space)  7.75-9 us/SYS\_GTOD (kernel space) |

**A note about concurrent Pausing**

If multiple processes (on separate devices) have triggered **E1** within **\epsilon** time bounded by **( ~ | E3-E1 | )**, **E2** may occur before **E1** has completed. This is because **E1-E2** is preemptable by the hardware interrupt. This is actually the correct behavior because **E2** should happen uniformly in both virtual time and wall clock time. After reaching **E3**, the device can return to its **E1** call which ensures that when resuming, the last device to finish its offline calculation is the one responsible for creating **E4**. This is because the hardware interrupts are on a uniform shared bus which ensures that the hardware interrupts are only triggered for the first occurence of a **HIGH** signal as well as the last occurence of returning to **LOW** signal. Additionally this design is highly scalable as each device only needs three exposed GPIO pins independant to the number of devices in the testbed.

TODO Verify implementation adheres to above design