What really happened on Mars?

Yajin Zhou (http://yajin.org)

Zhejiang University

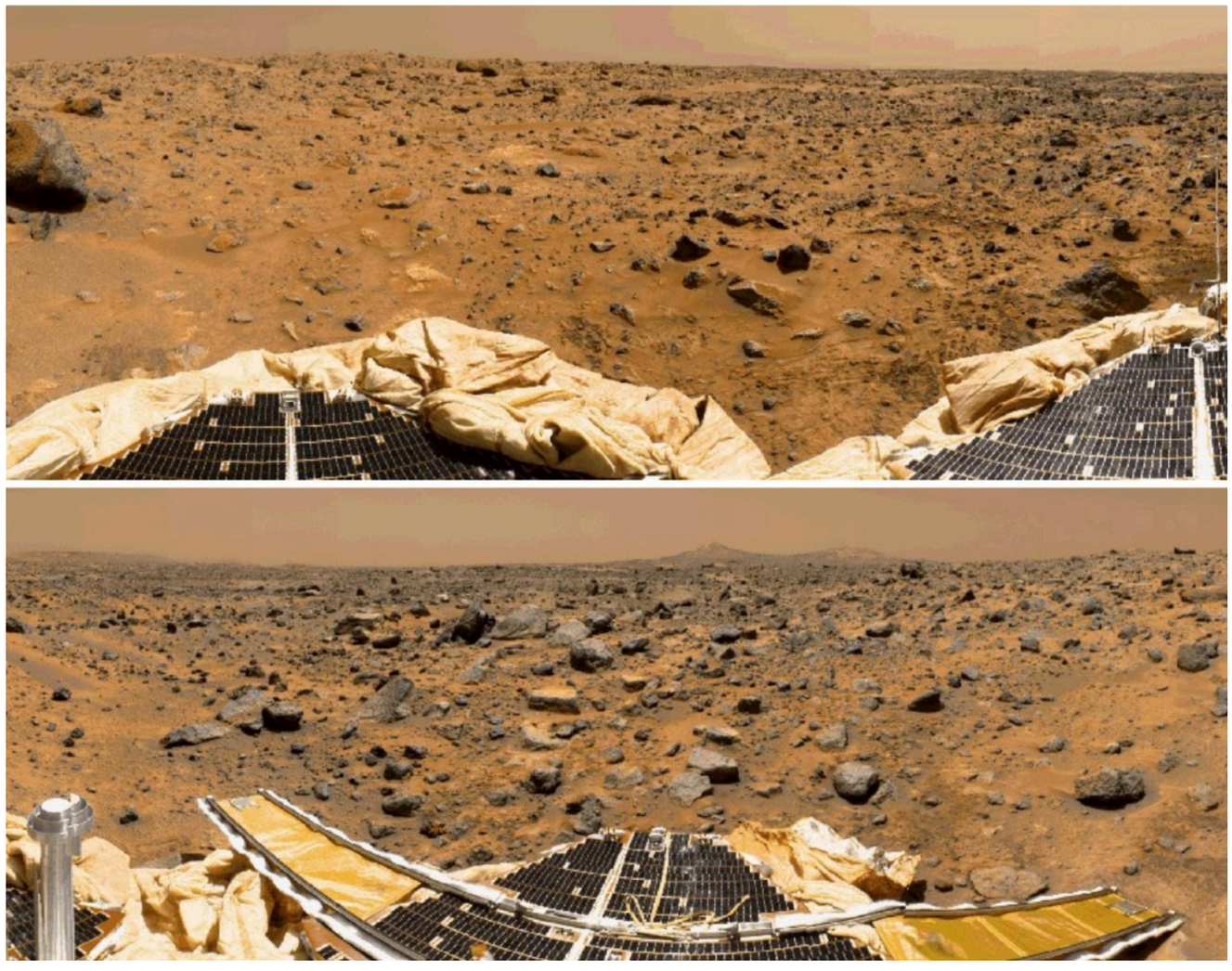
Sources:
http://itindex.net/detail/8003-%E7%81%AB%E6%98%9F
Priority Inversion on Mars by Jim Huang
http://research.microsoft.com/en-us/um/people/mbj/mars_pathfinder/mars_pathfinder.html

Mars Pathfinder

- Mission
  - Demonstrate new landing techniques: parachute and airbags
  - Take pictures
  - Analyze soil samples
- Demonstrate mobile robot technology: Sojourner

Sojourner Rover

Pictures taken



System Reset

- Within a few days of landing, when Pathfinder started gathering meteorological data, spacecraft began experiencing **total system resets**

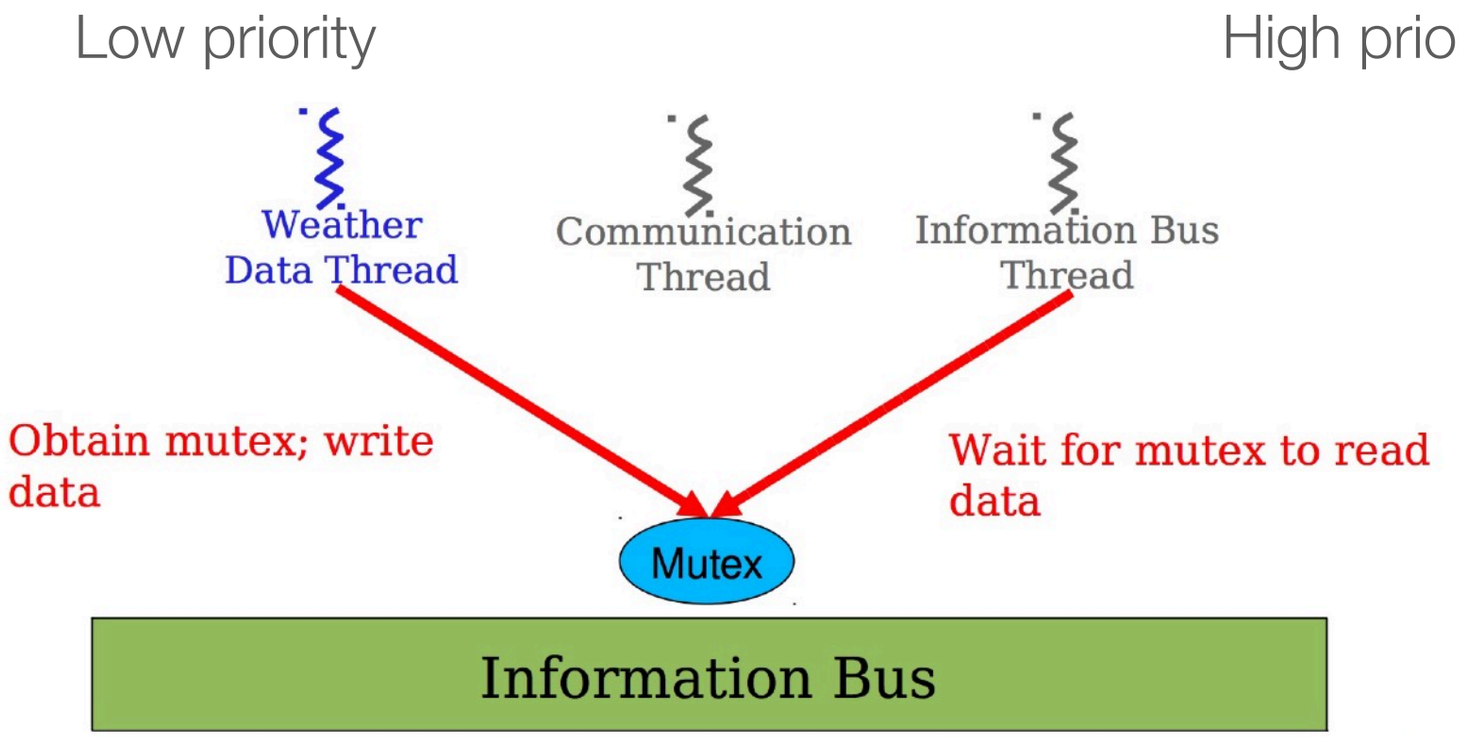
> Mary Beth Murrill, a spokeswoman for NASA's Jet Propulsion Laboratory, said transmission of the panoramic shot took “a lot of processing power.” She likened the data overload to what happens with a personal computer “when we ask it to do too many things at once.”
>
> The project manager, Brian Muirhead, said that to prevent a recurrence, controllers would schedule activities one after another, instead of at the same time. It was the second time the Pathfinder's computer had reset itself while trying to carry out several activities at once.
>
> In response, controllers reprogrammed the computer over the weekend to slow down the rate of activities and avoid another reset. But today, about an hour into a two-hour transmission session, it happened again.

System Reset

- This resulted in loss of data collected during each cycle
- JPL engineers had exact replica of the spacecraft in their lab
- They turned on the tracing feature of VxWorks
  - All system events such as context switches, uses of synchronization objects, and interrupts traced.
  - Tracing disabled on the actual spacecraft because generates too much data volume.
- After 18 hours of execution, early next morning when all but one engineer had gone home, the symptom was reproduced

VxWorks RTOS

- Multiple tasks, each with a priority
  - Higher priority tasks get to run before lower ones



VxWorks RTOS

- meteorological thread: When publishing its data, it would acquire a mutex, do writes to the bus, and release the mutex.

- If an interrupt caused the **information bus thread** to be scheduled while this mutex was held, and if the information bus thread then attempted to acquire this same mutex in order to retrieve published data, this would cause it to **block on the mutex**, waiting until the meteorological thread released the mutex before it could continue.

What happens if

- information bus thread is blocked on the mutex
- Interrupt occurs, and the medium priority thread is scheduled to run. And this thread runs for a long time
- During this period, the lower priority cannot run. Since it holds the mutex, then high information bus thread cannot run — it's waiting for the release of the mutex
- Watchdog will reset the machine

priority inversion

How to fix it

- When created, a VxWorks mutex object accepts a boolean parameter that indicates whether priority inheritance should be performed by the mutex. The mutex in question had been initialized with the parameter off; had it been on, the low-priority meteorological thread would have inherited the priority of the high-priority data bus thread blocked on it while it held the mutex, causing it be scheduled with higher priority than the medium-priority communications task, thus preventing the priority inversion. Once diagnosed, it was clear to the JPL engineers that using priority inheritance would prevent the resets they were seeing.

How to fix it

VxWorks contains a C language interpreter intended to allow developers to type in C expressions and functions to be executed on the fly during system debugging. The JPL engineers fortuitously decided to launch the spacecraft with this feature still enabled. By coding convention, the initialization parameter for the mutex in question (and those for two others which could have caused the same problem) were stored in global variables, whose addresses were in symbol tables also included in the launch software, and available to the C interpreter. A short C program was uploaded to the spacecraft, which when interpreted, changed the values of these variables from FALSE to TRUE. No more system resets occurred.

Lessons

- First and foremost, diagnosing this problem as a black box would have been impossible. Only detailed traces of actual system behavior enabled the faulty execution sequence to be captured and identified.

- Secondly, leaving the "debugging" facilities in the system saved the day. Without the ability to modify the system in the field, the problem could not have been corrected.

- Finally, the engineer's initial analysis that "the data bus task executes very frequently and is time-critical -- **we shouldn't spend the extra time in it to perform priority inheritance**" was exactly wrong. It is precisely in such time critical and important situations where **correctness is essential**, even at some additional performance cost.

Lessons

- David told us that the JPL engineers later **confessed that one or two system resets had occurred in their months of pre-flight testing**. They had never been reproducible or **explainable**, and so the engineers, **in a very human-nature response of denial**, decided that they probably weren't important, using the rationale "**it was probably caused by a hardware glitch**".

打破砂锅问到底