**Why Pthreads?**

**Light Weight:**

* When compared to the cost of creating and managing a process, a thread can be created with much less operating system overhead. Managing threads requires fewer system resources than managing processes.
* For example, the following table compares timing results for the **fork()** subroutine and the **pthread\_create()** subroutine. Timings reflect 50,000 process/thread creations, were performed with the **time** utility, and units are in seconds, no optimization flags.

Note: don't expect the sytem and user times to add up to real time, because these are SMP systems with multiple CPUs/cores working on the problem at the same time. At best, these are approximations run on local machines, past and present.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Platform** | **fork()** | | | **pthread\_create()** | | |
| **real** | **user** | **sys** | **real** | **user** | **sys** |
| **Intel 2.6 GHz Xeon E5-2670 (16 cores/node)** | 8.1 | 0.1 | 2.9 | 0.9 | 0.2 | 0.3 |
| **Intel 2.8 GHz Xeon 5660 (12 cores/node)** | 4.4 | 0.4 | 4.3 | 0.7 | 0.2 | 0.5 |
| **AMD 2.3 GHz Opteron (16 cores/node)** | 12.5 | 1.0 | 12.5 | 1.2 | 0.2 | 1.3 |
| **AMD 2.4 GHz Opteron (8 cores/node)** | 17.6 | 2.2 | 15.7 | 1.4 | 0.3 | 1.3 |
| **IBM 4.0 GHz POWER6 (8 cpus/node)** | 9.5 | 0.6 | 8.8 | 1.6 | 0.1 | 0.4 |
| **IBM 1.9 GHz POWER5 p5-575 (8 cpus/node)** | 64.2 | 30.7 | 27.6 | 1.7 | 0.6 | 1.1 |
| **IBM 1.5 GHz POWER4 (8 cpus/node)** | 104.5 | 48.6 | 47.2 | 2.1 | 1.0 | 1.5 |
| **INTEL 2.4 GHz Xeon (2 cpus/node)** | 54.9 | 1.5 | 20.8 | 1.6 | 0.7 | 0.9 |
| **INTEL 1.4 GHz Itanium2 (4 cpus/node)** | 54.5 | 1.1 | 22.2 | 2.0 | 1.2 | 0.6 |

==============================================================================

C Code for fork() creation test

==============================================================================

#include <stdio.h>

#include <stdlib.h>

#define NFORKS 50000

void do\_nothing() {

int i;

i= 0;

}

int main(int argc, char \*argv[]) {

int pid, j, status;

for (j=0; j<NFORKS; j++) {

/\*\*\* error handling \*\*\*/

if ((pid = fork()) < 0 ) {

printf ("fork failed with error code= %d\n", pid);

exit(0);

}

/\*\*\* this is the child of the fork \*\*\*/

else if (pid ==0) {

do\_nothing();

exit(0);

}

/\*\*\* this is the parent of the fork \*\*\*/

else {

waitpid(pid, status, 0);

}

}

}

==============================================================================

C Code for pthread\_create() test

==============================================================================

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#define NTHREADS 50000

void \*do\_nothing(void \*null) {

int i;

i=0;

pthread\_exit(NULL);

}

int main(int argc, char \*argv[]) {

int rc, i, j, detachstate;

pthread\_t tid;

pthread\_attr\_t attr;

pthread\_attr\_init(&attr);

pthread\_attr\_setdetachstate(&attr, PTHREAD\_CREATE\_JOINABLE);

for (j=0; j<NTHREADS; j++) {

rc = pthread\_create(&tid, &attr, do\_nothing, NULL);

if (rc) {

printf("ERROR; return code from pthread\_create() is %d\n", rc);

exit(-1);

}

/\* Wait for the thread \*/

rc = pthread\_join(tid, NULL);

if (rc) {

printf("ERROR; return code from pthread\_join() is %d\n", rc);

exit(-1);

}

}

pthread\_attr\_destroy(&attr);

pthread\_exit(NULL);

}