SPACECHAIN OS REAL-TIME PERFORMANCE

Test report

2018/05/12

Revision history		
Date	**Version No.**	**Revised contents**
12th May 2018	V1.0	New document

Contents

Test purpose

This test report is the real-time performance test report of SpaceChain OS, which intends to summarize the test during test phase and analyze the test result.

Test design

Introduction to the test OS

SpaceChain OS operation system

SpaceChain OS is originated from SylixOS. SylixOS is an embedded hard real-time operation system, and there are some famous similar operation systems in the world, such as VxWorks (mainly used in aeronautics and astronautics, military and industrial automation), RTEMS (stemming from the missile and rocket control real-time system in Pentagon), ThreadX (mainly for aeronautics and astronautics and digital communication). Globally, SpaceChain OS, as a late-comer of the real-time operation system, absorbs the designing thoughts of many real-time operation systems, including RTEMS, VxWorks and ThreadX, to enable it to catch up with or go beyond many other operation systems in specific performance parameters, and to become the top class of the domestic real-time operation systems.

Linux operation system

Linux is a set of operation system, free of charge, freely spread and similar to Unix. It conforms to POSIX standard for multi-user and multi-task, and supports 32 bits and 64 bits as well as multi-core CPU.

Linux+RT patch

The standard Linux kernel can only satisfy the requirement for soft real-time application, but it cannot guarantee the deadline. The Linux real-time preemptive patch PREEMPT_RT makes Linux have the hard real-time capability.

The Linux real-time preemptive patch drew attention in industrial circles: its concise design and the conformity with the kernel mainline enable it has a place in the fields of professional video and industrial control.

Introduction to test tools

Cyclictest

In this test, we use Cyclictest to carry out the real-time performance test. Cyclictest is

the most popular test tool in re-tests project. It is a test program with high accuracy, generally used for the test of operation system latency, to judge the real-time performance of an operation system.

The principle of Cyclictest will be analyzed with the source code below.

The main function of rt-tests/src/cyclictest/cyclictest.c is shown in manifest 2.1.

Manifest 21 cyclictest main function

```
int main(int argc, char **argv)
{       ...
        stat->min = 1000000;
         stat->max = 0;
         stat->avg = 0.0;
         stat->threadstarted = 1;
         status = pthread_create(&stat->thread, &attr, timerthread, par);
          ...
}
```

The main function only initializes correlated variables, but the specific test is done by the created thread. The test and the record are done in the test thread which is shown in the manifest 2.2.

Manifest 2.2 cyclictest test thread

```
void *timerthread(void *param)
{       ...
       interval.tv_sec = par->interval                    // assign the interval in the parameter to the interval
in the function
       interval.tv_nsec = (par->interval % USEC_PER_SEC) * 1000;
        ...
       /* Get current time    */
       clock_gettime(par->clock, &now);                        // get current time, and save it in now


       next = now;                                                 // these three lines have the current time
(value of now)
       next.tv_sec += interval.tv_sec;                    // be added with the interval
       next.tv_nsec += interval.tv_nsec;                       // work out the nest interval, save it in next
       tsnorm(&next);
        ...
        /* Wait for next period */                          // Wait for next period
               ...
```

```
        if ((ret = clock_gettime(par->clock, &now))) {      // save the time of the next period in now, and at this moment the number saved in the value of now is the real value of the next period, whereas the above value saved in next is the value of the previous period added with the interval, and is also, theoretically, the value of the next period
                if (ret != EINTR)
                    warn("clock_getttime() failed. errno: %d\n", errno);
                goto out;
            }
        if (use_nsecs)
            diff = calcdiff_ns(now, next);          // as mentioned above, in now is the real value of the next period, but next is the theoretical value, so the difference between these two is the latency, which assigns value to diff
        else
            diff = calcdiff(now, next);
        if (diff < stat->min)                       // if the latency is smaller than min, change min to this diff of smaller latency value
            stat->min = diff;
        if (diff > stat->max) {                     // if the latency is greater than max, change max to this diff of greater latency value
            stat->max = diff;
            if (refresh_on_max)
                pthread_cond_signal(&refresh_on_max_cond);
        }
        stat->avg += (double) diff;                 // work out the new average latency
         ...
        /* Update the histogram */                  // update the latency statistic data saved in histogram
        if (histogram) {
          if (diff >= histogram) {                  // if the latency is greater than histogram, add one overflow
                stat->hist_overflow++;
                                    if (stat->num_outliers < histogram)
                    stat->outliers[stat->num_outliers++] = stat->cycles;
            }
            else                                    // if there is no overflow, add 1 to corresponding value in histogram
                stat->hist_array[diff]++;
        }

        stat->cycles++;                         // add 1 to cycles
```

```
    next.tv_sec += interval.tv_sec;          // continue calculating the value of the next cycle...
    next.tv_nsec += interval.tv_nsec;
     ...
}
```

Cyclictest makes loop scheduling by test thread, sets the interval of scheduling, acquires the latency in actual operation by comparing the difference between the real time and the calculated time, and judges the real-time performance of the operation system by the value of the latency.

Hackbench

Hackbench is a reference test program to test performance, overhead and scalability of the Linux process scheduler. Its test can run on different hardware platforms and in various Linux OS versions, by simulating the communication between the client process and the server process of a set of C/S mode to test performance, overhead and scalability of the Linux process scheduler.

Hackbench test is divided into groups which number is given by the command line, and each group has num fds groups read-write processes. The default value of num fds is 20. In each test group, num fds client processes (reading process) and num fds server processes (writing process) created are respectively responsible for information reading and writing, and num fds pipelines created are for communication between the reading and writing processes. The writing process writes data for the pipeline, while the reading process reads data in the pipeline.

By this, it can keep communicating between processes and create processes continuously, so as to provide Cyclictest test program with a pressured situation.

Test environment and configuration

Test environment

The test environment is shown in Table 2.1.

Table 2.1 . Test environment

Software and hardware	Model
Single-core test development board	OK335xS
Multi-core test development board	E9
Serial port output client	PuttY

Corresponding test operation system configuration

Corresponding test operation system configuration is shown in Table 2.2.

Table 2.2 Corresponding test operation system configuration

Operation system	Version
SpaceChain OS operation system	STABLE-1.1.1
OK335xS with Linux operation system	3.2.0
Linux operation system with RT patch	3.2.0-rt10
E9 with Linux operation system	Linux EmbedSky 3.0.35

Test method

Single-core test

In the single-core test, we use OK335xS development board to test, and in this board is the Linux operation system which kernel version number is 3.2.0. The real-time test, based on this, is to evaluate the real-time performance difference between SpaceChain OS and Linux as well as Linux+RT, mainly including the following two tests:

1. Non-pressure test:

Run 1000000 cycles of cyclictest test in Linux, Linux+RT and SpaceChain OS, with interval of two cycles 10ms.

2. Pressured test

Run 1000000 cycles of cyclictest test in Linux, Linux+RT and SpaceChain OS, with interval of two cycles 10ms. At the same time, run hackbench to provide pressured situation, so as to complete the pressured test.

Multi-core test

In multi-core test, we use E9 development board to test, and in this board is the Linux operation system which kernel version number is 3.0.35.

The test method is still use cyclictest and hackbench to test.

When Linux in this board is installed RT patch, it cannot be normally started, so this test can only be performed in SpaceChain OS and in Linux operation system matched with it. It also is divided into non-pressure and pressured tests, and compared with the real-time performance in SpaceChain OS single-core.

Test

Test process

Test range and requirement

The test range includes: single-core pressured test, non-pressure test, multi-core non-pressure test and multi-core pressure test.

Test time

The test time is shown in Table 3.1.

Table 3.1 Test time

Test type	Test time
Single-core Linux non-pressure test	30th June, 2015 09:00-12:00
Single-core SpaceChain OS non-pressure test	30th June, 2015 13:00-16:00
Single-core Linux+RT non-pressure test	30th June, 2015 17:00-20:00
Single-core Linux pressured test	30th June, 2015 20:00-21:00
Single-core SpaceChain OS pressured test	30th June, 2015 21:00-22:00
Single-core Linux+RT pressured test	30th June, 2015 22:00-23:00
Mulit-core SpaceChain OS and Linux test	1st July, 2015 09:00-10:00

Test institute and personnel

This test is done by Dr. Chen Yizhang, Tsinghua University.

Test result statistics

Single-core non-pressure result statistics

- SpaceChain OS：

Single-core non-pressure test results of 1 million times are shown in Fig.3.1.

```
T: 0 (67174458) P: 2 I:10000 C:1000000 Min:      3 Act:    4 Avg:    4 Max:      35
```

Fig. 3.1 SpaceChain OS Single-core non-pressure test results of 1 million times

Test results are in the following table:

Max latency	35us
Min latency	3us
Average latency	4us

◆ Linux：

Single-core non-pressure test results of 1 million times are shown in Fig.3.2.

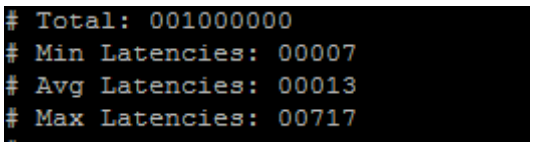

Fig. 3.2 Linux Single-core non-pressure test results of 1 million times

Test results are in the following table:

Max latency	717us
Min latency	7us
Average latency	13us

◆ Linux+RT patch:

Single-core non-pressure test results of 1 million times are shown in Fig.3.3.

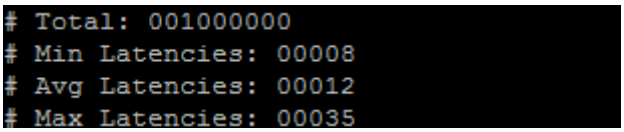

Fig. 3.3 Linux+RT Single-core non-pressure test results of 1 million times

Test results are in the following table:

Max latency	35us
Min latency	8us
Average latency	12us

Single-core pressured result statistics

◆ SpaceChain OS：

Single-core pressured test results of 100 thousand times are shown in Fig.3.4.

Fig. 3.4 SpaceChain OS single-core pressured test results of 100 thousand times

Test results are in the following table:

Max latency	62us
Min latency	3us

Average latency	4us

◆ Linux:

Single-core pressured test results of 100 thousand times are shown in Fig.3.5.

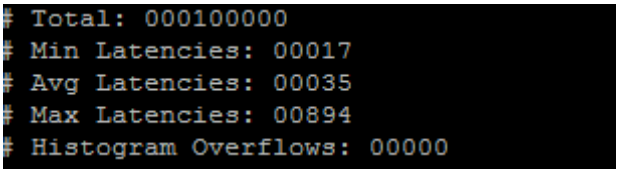

Fig. 3.5 Linux single-core pressured test results of 100 thousand times

Test results are in the following table:

Max latency	894us
Min latency	17us
Average latency	35us

◆ Linux+RT :

Single-core pressured test results of 100 thousand times are shown in Fig.3.6.

Fig. 3.6 Linux+RT single-core pressured test results of 100 thousand times

Test results are in the following table:

Max latency	67us
Min latency	8us
Average latency	31us

◆ SpaceChain OS：

Multi-core non-pressure test results of 100 thousand times are shown in Fig.3.7.

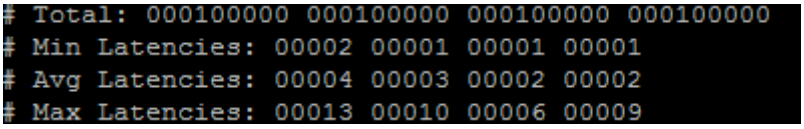

Fig. 3.7 SpaceChain OS multi-core non-pressure test results of 100 thousand times

Test results are in the following table:

	CPU#0	**CPU#1**	**CPU#2**	**CPU#3**
Min latency	2us	1us	1us	1us

Average latency	4us	3us	2us	2us
Max latency	13us	10us	6us	9us

◆ Linux：

Multi-core non-pressure test results of 100 thousand times are shown in Fig.3.8.

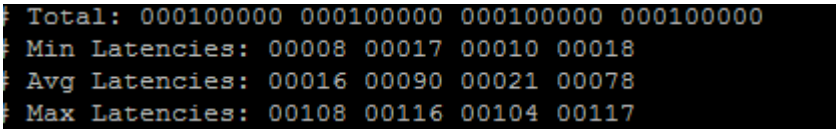

Fig. 3.8 Linux multi-core non-pressure test results of 100 thousand times

Test results are in the following table:

	CPU#0	CPU#1	CPU#2	CPU#3
Min latency	8us	17us	10us	18us
Average latency	16us	90us	21us	78us
Max latency	108us	116us	104us	117us

◆ SpaceChain OS：

Multi-core pressured test results of 100 thousand times are shown in Fig.3.9.

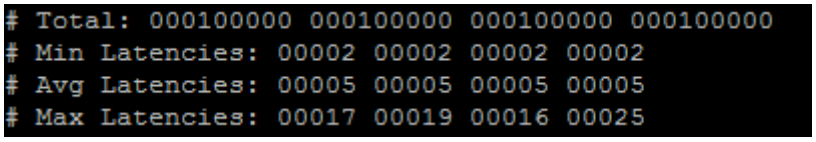

Fig. 3.9 SpaceChain OS multi-core pressured test results of 100 thousand times

Test results are in the following table:

	CPU#0	CPU#1	CPU#2	CPU#3
Min latency	2us	2us	2us	2us
Average latency	5us	5us	5us	5us
Max latency	17us	19us	16us	25us

◆ Linux：

Multi-core pressured test results of 100 thousand times are shown in Fig.3.10.

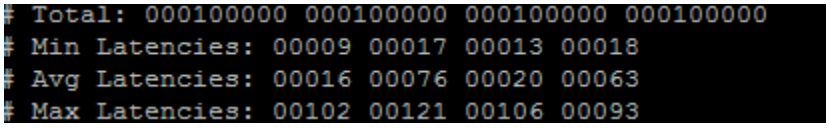

Fig. 3.10 Linux multi-core pressured test results of 100 thousand times

Test results are in the following table:

	CPU#0	CPU#1	CPU#2	CPU#3
Min latency	9us	17us	13us	18us
Average latency	16us	76us	20us	63us
Max latency	102us	121us	106us	93us

Test result analysis

Single-core non-pressure result analysis

First of all, it can be concluded from the statistics result that the real-time performance of Linux operation system without real-time patch is the poorest, which max latency is as high as 717μs, so the focus will be laid on SpaceChain OS and Linux+RT in the following analysis. Below is the statistical diagram of non-pressure latencies of 1 million times recorded by cyclictest tool, as shown in Fig. 3.11.

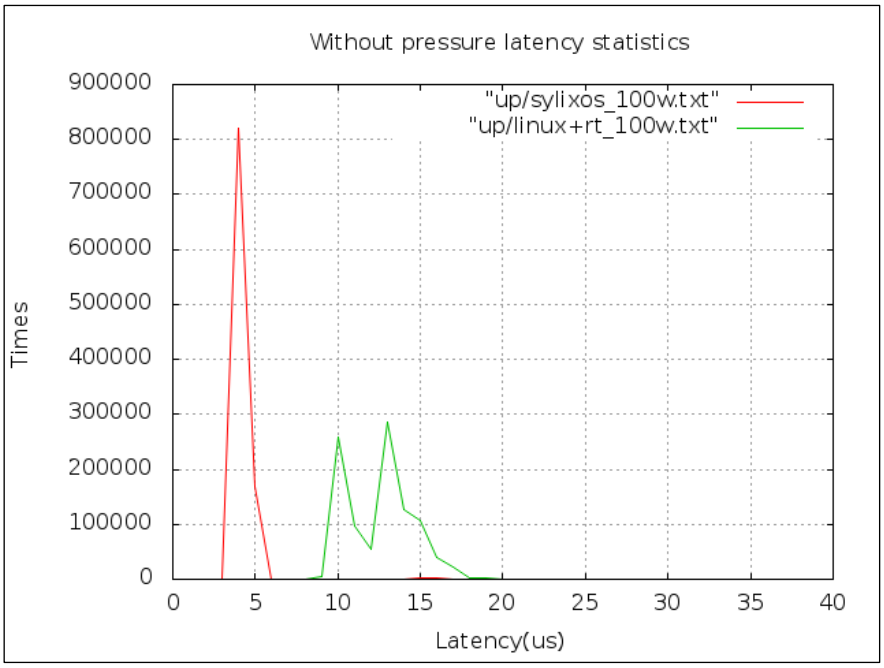

Fig. 3.11 Statistics of SpaceChain OS and Linux+RT single-core non-pressure latencies of 1 million times

From the statistical diagram, it can be seen that in the single-core non-pressure tests of 1 million times the maximum latencies of SpaceChain OS and Linux+RT are the same, which are 35μs; however, the average latencies of SpaceChain OS is relatively low, most of which are around 4μs, while those of Linux+RT are 13μs.

Comparison results are in the following table:

	SpaceChain OS	Linux+RT	Linux	Comparison results
Max latency (μs)	35	35	717	SpaceChain OS=Linux+RT<linux
Min latency (μs)	3	8	7	SpaceChain OS<linux<Linux+RT
Average latency (μs)	4	12	13	SpaceChain OS<Linux+RT<linux

Single-core pressured result analysis

First of all, it can be concluded from the statistics result that the real-time performance of Linux operation system without real-time patch is the poorest, which max latency is as high as 894μs, so the focus will be laid on SpaceChain OS and Linux+RT in the following analysis. Below is the statistical diagram of pressured latencies of 100 thousand times recorded by cyclictest tool, as shown in Fig. 3.12.

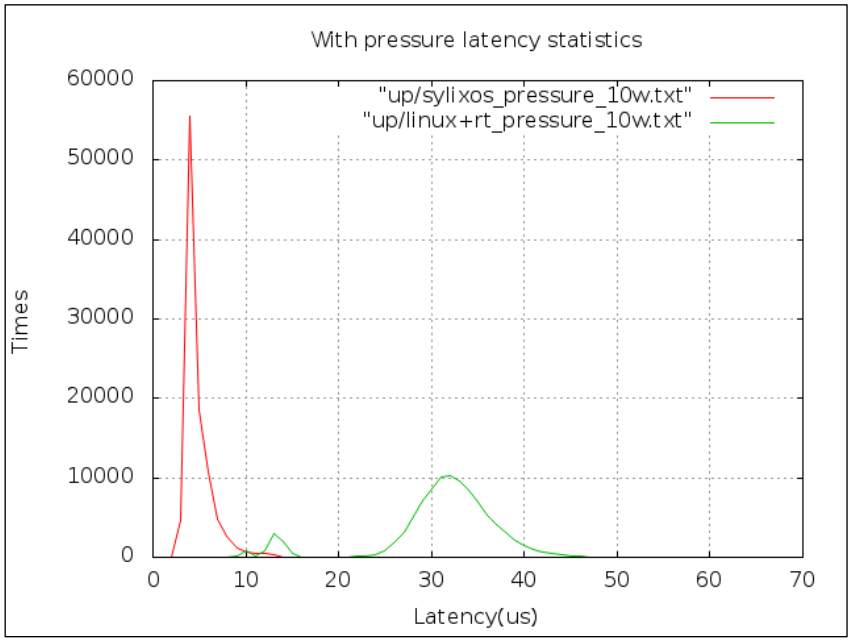

Fig. 3.12 Statistics of SpaceChain OS and Linux+RT single-core pressured latencies of 100 thousand times

From the statistical diagram, it can be known that in single-core pressured tests of 100 thousand times the maximum latencies of SpaceChain OS and Linux+RT increased compared with non-pressure condition. The max latency of SpaceChain OS is 62μs, whereas it of Linux+RT 67μs. The max latency of SpaceChain OS is lower, and from the statistical diagram it can be seen the average latency under pressured condition of SpaceChain OS is lower, too.

Comparison results are in the following table:

	SpaceChain OS	Linux+RT	Linux	Comparison results
Max latency (μs)	62	67	894	SpaceChain OS < Linux+RT < Linux
Min latency (μs)	2	8	17	SpaceChain OS < Linux+RT < Linux
Average latency (μs)	4	31	35	SpaceChain OS < Linux+RT < Linux

Multi-core non-pressure result analysis

Comparison results are in the following table:

	SpaceChain OS	Linux	Comparison results
Max latency (μs)	13	117	SpaceChain OS << Linux
Min latency (μs)	2	18	SpaceChain OS < Linux
Average latency (μs)	4	90	SpaceChain OS < Linux

From statistical result, in non-pressure condition the maximum latency of SpaceChain OS is apparently lower than it of Linux, and so are the minimum and average latencies.

Multi-core pressured result analysis

Comparison results are in the following table:

	SpaceChain OS	Linux	Comparison results
Max latency (μs)	25	121	SpaceChain OS << Linux
Min latency (μs)	2	18	SpaceChain OS < Linux
Average latency (μs)	5	76	SpaceChain OS < Linux

From statistical result, in pressured condition the maximum latency of SpaceChain OS is still lower than it of Linux, and so are the minimum and average latencies.

SpaceChain OS single- and multi-core results analysis

Comparison results of SpaceChain OS latencies in single- and multi-core non-pressure condition are shown in the following table:

	SpaceChain OS single-core	SpaceChain OS multi-core

Max latency (μs)	35	13
Min latency (μs)	3	2
Average latency (μs)	4	4

Comparison results of SpaceChain OS latencies in single- and multi-core pressured condition are shown in the following table:

	SpaceChain OS single-core	**SpaceChain OS multi-core**
Max latency (μs)	62	25
Min latency (μs)	2	2
Average latency (μs)	4	5

From the result, the max latency dose not decrease by the increase of CPU cores, even lower than the max latency in single-core, and the average time is almost the same. This is because the single-core performance of multi-core development board CPU is better than single-core board.

Test conclusion

This test mainly completes the real-time performance test of SpaceChain OS, Linux and Linux+RT in the single-core and the multi-core development boards.

In single-core non-pressure condition, SpaceChain OS has the lowest latency, and its max latency is same as that of Linux+RT, both 35μs. It means that the real-time performance of Linux+RT is almost same as it of SpaceChain OS, while the average latency of SpaceChain OS is better than it of Linux+RT.

In single-core pressured condition, the average latency of SpaceChain OS is still the lowest among these three systems, and the max latency of it is lower than that of Linux+RT, which means the real-time performance of SpaceChain OS with pressure is better than it of Linux+RT.

In multi-core condition, the real-time performances of SpaceChain OS both with and without pressure are better than those of Linux operation system. And by comparison with the real-time performance of SpaceChain OS in single-core condition, it is found that the multi-core real-time performance will not be adversely affected by the increase of the CPU cores.