Block io priorities

Intro

With the introduction of cfq v3 (aka cfq-ts or time sliced cfq), basic io priorities are supported for reads on files. This enables users to io nice processes or process groups, similar to what has been possible with cpu scheduling for ages. This document mainly details the current possibilities with cfq; other io schedulers do not support io priorities thus far.

Scheduling classes

CFQ implements three generic scheduling classes that determine how io is served for a process.

IOPRIO_CLASS_RT: This is the realtime io class. This scheduling class is given higher priority than any other in the system, processes from this class are given first access to the disk every time. Thus it needs to be used with some care, one io RT process can starve the entire system. Within the RT class, there are 8 levels of class data that determine exactly how much time this process needs the disk for on each service. In the future this might change to be more directly mappable to performance, by passing in a wanted data rate instead.

IOPRIO_CLASS_BE: This is the best-effort scheduling class, which is the default for any process that hasn't set a specific io priority. The class data determines how much io bandwidth the process will get, it's directly mappable to the cpu nice levels just more coarsely implemented. 0 is the highest BE prio level, 7 is the lowest. The mapping between cpu nice level and io nice level is determined as: io_nice = (cpu_nice + 20) / 5.

IOPRIO_CLASS_IDLE: This is the idle scheduling class, processes running at this level only get io time when no one else needs the disk. The idle class has no class data, since it doesn't really apply here.

Tools

See below for a sample ionice tool. Usage:

```
# ionice -c<class> -n<level> -p<pid>
```

If pid isn't given, the current process is assumed. IO priority settings are inherited on fork, so you can use ionice to start the process at a given level:

```
# ionice -c2 -n0 /bin/ls
```

will run ls at the best-effort scheduling class at the highest priority. For a running process, you can give the pid instead:

```
# ionice -c1 -n2 -p100
```

will change pid 100 to run at the realtime scheduling class, at priority 2.

ionice.c tool:

```
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <getopt.h>
#include <unistd.h>
#include <sys/ptrace.h>
#include <asm/unistd.h>

extern int sys_ioprio_set(int, int, int);
extern int sys_ioprio_get(int, int);

#if defined(__i386__)
#define __NR_ioprio_set		289
#define __NR_ioprio_get		290
#elif defined(__ppc__)
#define __NR_ioprio_set		273
#define __NR_ioprio_get		274
#elif defined(__x86_64__)
#define __NR_ioprio_set		251
#define __NR_ioprio_get		252
#elif defined(__ia64__)
#define __NR_ioprio_set		1274
#define __NR_ioprio_get		1275
#else
#error "Unsupported arch"
#endif

static inline int ioprio_set(int which, int who, int ioprio)
{
	return syscall(__NR_ioprio_set, which, who, ioprio);
}
```

```
static inline int ioprio_get(int which, int who)
{
	return syscall(__NR_ioprio_get, which, who);
}

enum {
	IOPRIO_CLASS_NONE,
	IOPRIO_CLASS_RT,
	IOPRIO_CLASS_BE,
	IOPRIO_CLASS_IDLE,
};

enum {
	IOPRIO_WHO_PROCESS = 1,
	IOPRIO_WHO_PGRP,
	IOPRIO_WHO_USER,
};

#define IOPRIO_CLASS_SHIFT	13

const char *to_prio[] = { "none", "realtime", "best-effort", "idle", };

int main(int argc, char *argv[])
{
	int ioprio = 4, set = 0, ioprio_class = IOPRIO_CLASS_BE;
	int c, pid = 0;

	while ((c = getopt(argc, argv, "+n:c:p:")) != EOF) {
		switch (c) {
		case 'n':
			ioprio = strtol(optarg, NULL, 10);
			set = 1;
			break;
		case 'c':
			ioprio_class = strtol(optarg, NULL, 10);
			set = 1;
			break;
		case 'p':
			pid = strtol(optarg, NULL, 10);
			break;
		}
	}

	switch (ioprio_class) {
		case IOPRIO_CLASS_NONE:
			ioprio_class = IOPRIO_CLASS_BE;
			break;
		case IOPRIO_CLASS_RT:
		case IOPRIO_CLASS_BE:
			break;
		case IOPRIO_CLASS_IDLE:
			ioprio = 7;
			break;
		default:
			printf("bad prio class %d\n", ioprio_class);
			return 1;
	}

	if (!set) {
		if (!pid && argv[optind])
			pid = strtol(argv[optind], NULL, 10);

		ioprio = ioprio_get(IOPRIO_WHO_PROCESS, pid);

		printf("pid=%d, %d\n", pid, ioprio);

		if (ioprio == -1)
			perror("ioprio_get");
		else {
			ioprio_class = ioprio >> IOPRIO_CLASS_SHIFT;
			ioprio = ioprio & 0xff;
			printf("%s: prio %d\n", to_prio[ioprio_class], ioprio);
		}
	} else {
		if (ioprio_set(IOPRIO_WHO_PROCESS, pid, ioprio | ioprio_class << IOPRIO_CLASS_SHIFT) == -1)
			perror("ioprio_set");
			return 1;
		}

		if (argv[optind])
			execvp(argv[optind], &argv[optind]);
	}

	return 0;
```

```
}
```

March 11 2005, Jens Axboe <jens.axboe@oracle.com>