

```

*****
36288 Fri Jan 15 13:15:02 2016
new/usr/src/uts/common/io/cpudrv.c
XXXX cpudrv attach is racy
*****
_____unchanged_portion_omitted_____

235 /*
236  * Driver attach(9e) entry point.
237  */
238 static int
239 cpudrv_attach(dev_info_t *dip, ddi_attach_cmd_t cmd)
240 {
241     int             instance;
242     cpudrv_devstate_t *cpudsp;

244     instance = ddi_get_instance(dip);

246     switch (cmd) {
247     case DDI_ATTACH:
248         DPRINTF(D_ATTACH, ("cpudrv_attach: instance %d: "
249             "DDI_ATTACH called\n", instance));
250         if (!cpudrv_is_enabled(NULL))
251             return (DDI_FAILURE);
252         if (ddi_soft_state_zalloc(cpudrv_state, instance) !=
253             DDI_SUCCESS) {
254             cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
255                 "can't allocate state", instance);
256             cpudrv_enabled = B_FALSE;
257             return (DDI_FAILURE);
258         }
259         if ((cpudsp = ddi_get_soft_state(cpudrv_state, instance)) ==
260             NULL) {
261             cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
262                 "can't get state", instance);
263             ddi_soft_state_free(cpudrv_state, instance);
264             cpudrv_enabled = B_FALSE;
265             return (DDI_FAILURE);
266         }
267         cpudsp->dip = dip;

269         /*
270          * Find CPU number for this dev_info node.
271          */
272         if (!cpudrv_get_cpu_id(dip, &(cpudsp->cpu_id))) {
273             cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
274                 "can't convert dip to cpu_id", instance);
275             ddi_soft_state_free(cpudrv_state, instance);
276             cpudrv_enabled = B_FALSE;
277             return (DDI_FAILURE);
278         }

280         if (!cpudrv_is_enabled(cpudsp)) {
281             cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
282                 "not supported or it got disabled on us",
283                 instance);
284             cpudrv_enabled = B_FALSE;
285             ddi_soft_state_free(cpudrv_state, instance);
286             return (DDI_FAILURE);
287         }

289 #endif /* ! codereview */
290     mutex_init(&cpudsp->lock, NULL, MUTEX_DRIVER, NULL);
291     if (cpudrv_init(cpudsp) != DDI_SUCCESS) {
292         cpudrv_enabled = B_FALSE;

```

```

293     cpudrv_free(cpudsp);
294     ddi_soft_state_free(cpudrv_state, instance);
295     return (DDI_FAILURE);
296 }
297 if (cpudrv_comp_create(cpudsp) != DDI_SUCCESS) {
298     cpudrv_enabled = B_FALSE;
299     cpudrv_free(cpudsp);
300     ddi_soft_state_free(cpudrv_state, instance);
301     return (DDI_FAILURE);
302 }
303 if (ddi_prop_update_string(DDI_DEV_T_NONE,
304     dip, "pm-class", "CPU") != DDI_PROP_SUCCESS) {
305     cpudrv_enabled = B_FALSE;
306     cpudrv_free(cpudsp);
307     ddi_soft_state_free(cpudrv_state, instance);
308     return (DDI_FAILURE);
309 }

311 /*
312  * Taskq is used to dispatch routine to monitor CPU
313  * activities.
314  */
315 cpudsp->cpudrv_pm.tq = ddi_taskq_create(dip,
316     "cpudrv_monitor", CPUDRV_TASKQ_THREADS,
317     TASKQ_DEFAULTPRI, 0);

319 mutex_init(&cpudsp->cpudrv_pm.timeout_lock, NULL,
320     MUTEX_DRIVER, NULL);
321 cv_init(&cpudsp->cpudrv_pm.timeout_cv, NULL,
322     CV_DEFAULT, NULL);

324 /*
325  * Driver needs to assume that CPU is running at
326  * unknown speed at DDI_ATTACH and switch it to the
327  * needed speed. We assume that initial needed speed
328  * is full speed for us.
329  */
330 /*
331  * We need to take the lock because cpudrv_monitor()
332  * will start running in parallel with attach().
333  */
334 mutex_enter(&cpudsp->lock);
335 cpudsp->cpudrv_pm.cur_spd = NULL;
336 cpudsp->cpudrv_pm.pm_started = B_FALSE;
337 /*
338  * We don't call pm_raise_power() directly from attach
339  * because driver attach for a slave CPU node can
340  * happen before the CPU is even initialized. We just
341  * start the monitoring system which understands
342  * unknown speed and moves CPU to top speed when it
343  * has been initialized.
344  */
345 CPUDRV_MONITOR_INIT(cpudsp);
346 mutex_exit(&cpudsp->lock);

338 }

348 if (!cpudrv_mach_init(cpudsp)) {
349     cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
350         "cpudrv_mach_init failed", instance);
351     cpudrv_enabled = B_FALSE;
352     cpudrv_free(cpudsp);
353     ddi_soft_state_free(cpudrv_state, instance);
354     return (DDI_FAILURE);
355 }

```

```
357         CPUDRV_INSTALL_MAX_CHANGE_HANDLER(cpudsp);
359         (void) ddi_prop_update_int(DDI_DEV_T_NONE, dip,
360             DDI_NO_AUTODETACH, 1);
361         ddi_report_dev(dip);
362         return (DDI_SUCCESS);
364     case DDI_RESUME:
365         DPRINTF(D_ATTACH, ("cpudrv_attach: instance %d: "
366             "DDI_RESUME called\n", instance));
368         cpudsp = ddi_get_soft_state(cpudrv_state, instance);
369         ASSERT(cpudsp != NULL);
371         /*
372          * Nothing to do for resume, if not doing active PM.
373          */
374         if (!cpudrv_is_enabled(cpudsp))
375             return (DDI_SUCCESS);
377         mutex_enter(&cpudsp->lock);
378         /*
379          * Driver needs to assume that CPU is running at unknown speed
380          * at DDI_RESUME and switch it to the needed speed. We assume
381          * that the needed speed is full speed for us.
382          */
383         cpudsp->cpudrv_pm.cur_spd = NULL;
384         CPUDRV_MONITOR_INIT(cpudsp);
385         mutex_exit(&cpudsp->lock);
386         CPUDRV_REDEFINE_TOPSPEED(dip);
387         return (DDI_SUCCESS);
389     default:
390         return (DDI_FAILURE);
391     }
392 }
```

unchanged\_portion\_omitted