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

```
/*
 * Driver attach(9e) entry point.
 */
static int
cpudrv_attach(dev_info_t *dip, ddi_attach_cmd_t cmd)
{
        int                     instance;
        cpudrv_devstate_t       *cpudsp;

        instance = ddi_get_instance(dip);

        switch (cmd) {
        case DDI_ATTACH:
                DPRINTF(D_ATTACH, ("cpudrv_attach: instance %d: "
                    "DDI_ATTACH called\n", instance));
                if (!cpudrv_is_enabled(NULL))
                        return (DDI_FAILURE);
                if (ddi_soft_state_zalloc(cpudrv_state, instance) !=
                    DDI_SUCCESS) {
                        cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
                            "can’t allocate state", instance);
                        cpudrv_enabled = B_FALSE;
                        return (DDI_FAILURE);
                }
                if ((cpudsp = ddi_get_soft_state(cpudrv_state, instance)) ==
                    NULL) {
                        cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
                            "can’t get state", instance);
                        ddi_soft_state_free(cpudrv_state, instance);
                        cpudrv_enabled = B_FALSE;
                        return (DDI_FAILURE);
                }
                cpudsp->dip = dip;

                /*
                 * Find CPU number for this dev_info node.
                 */
                if (!cpudrv_get_cpu_id(dip, &(cpudsp->cpu_id))) {
                        cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
                            "can’t convert dip to cpu_id", instance);
                        ddi_soft_state_free(cpudrv_state, instance);
                        cpudrv_enabled = B_FALSE;
                        return (DDI_FAILURE);
                }

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

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

```
                        cpudrv_free(cpudsp);
                        ddi_soft_state_free(cpudrv_state, instance);
                        return (DDI_FAILURE);
                }
                if (cpudrv_comp_create(cpudsp) != DDI_SUCCESS) {
                        cpudrv_enabled = B_FALSE;
                        cpudrv_free(cpudsp);
                        ddi_soft_state_free(cpudrv_state, instance);
                        return (DDI_FAILURE);
                }
                if (ddi_prop_update_string(DDI_DEV_T_NONE,
                    dip, "pm-class", "CPU") != DDI_PROP_SUCCESS) {
                        cpudrv_enabled = B_FALSE;
                        cpudrv_free(cpudsp);
                        ddi_soft_state_free(cpudrv_state, instance);
                        return (DDI_FAILURE);
                }

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

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

                /*
                 * Driver needs to assume that CPU is running at
                 * unknown speed at DDI_ATTACH and switch it to the
                 * needed speed. We assume that initial needed speed
                 * is full speed for us.
                 */
                /*
                 * We need to take the lock because cpudrv_monitor()
                 * will start running in parallel with attach().
                 */
                mutex_enter(&cpudsp->lock);
                cpudsp->cpudrv_pm.cur_spd = NULL;
                cpudsp->cpudrv_pm.pm_started = B_FALSE;
                /*
                 * We don’t call pm_raise_power() directly from attach
                 * because driver attach for a slave CPU node can
                 * happen before the CPU is even initialized. We just
                 * start the monitoring system which understands
                 * unknown speed and moves CPU to top speed when it
                 * has been initialized.
                 */
                CPUDRV_MONITOR_INIT(cpudsp);
                mutex_exit(&cpudsp->lock);

                }

                if (!cpudrv_mach_init(cpudsp)) {
                        cmn_err(CE_WARN, "cpudrv_attach: instance %d: "
                            "cpudrv_mach_init failed", instance);
                        cpudrv_enabled = B_FALSE;
                        cpudrv_free(cpudsp);
                        ddi_soft_state_free(cpudrv_state, instance);
                        return (DDI_FAILURE);
                }
```

```
		CPUDRV_INSTALL_MAX_CHANGE_HANDLER(cpudsp);

		(void) ddi_prop_update_int(DDI_DEV_T_NONE, dip,
		    DDI_NO_AUTODETACH, 1);
		ddi_report_dev(dip);
		return (DDI_SUCCESS);

	case DDI_RESUME:
		DPRINTF(D_ATTACH, ("cpudrv_attach: instance %d: "
		    "DDI_RESUME called\n", instance));

		cpudsp = ddi_get_soft_state(cpudrv_state, instance);
		ASSERT(cpudsp != NULL);

		/*
		 * Nothing to do for resume, if not doing active PM.
		 */
		if (!cpudrv_is_enabled(cpudsp))
			return (DDI_SUCCESS);

		mutex_enter(&cpudsp->lock);
		/*
		 * Driver needs to assume that CPU is running at unknown speed
		 * at DDI_RESUME and switch it to the needed speed. We assume
		 * that the needed speed is full speed for us.
		 */
		cpudsp->cpudrv_pm.cur_spd = NULL;
		CPUDRV_MONITOR_INIT(cpudsp);
		mutex_exit(&cpudsp->lock);
		CPUDRV_REDEFINE_TOPSPEED(dip);
		return (DDI_SUCCESS);

	default:
		return (DDI_FAILURE);
	}
}
```
______***unchanged_portion_omitted_***