Päätoimittajan palsta

Tuleeko kaikista rinnakkaisohjelmoijia?

Maailman varmaankin tunnetuin mikroprosessorivalmistaja Intel järjesti cocktailtilaisuuden eurooppalaisen teoreettisen ohjelmistotieteen päätapahtuman ETAPS 2006 (The European Joint Conferences on Theory and Practice of Software 2006) yhteydessä maaliskuun lopulla Wienissä. Kutsuttuja olivat kaikki konferenssirykelmän sadat osallistujat. Intel halusi kertoa heille kaksi asiaa.

Ensimmäisen viestin olin kuullut jo muuta kautta. Mikroprosessorien tehon hurjaa kasvua kuvaava Mooren laki on taitekohdassa. Intel ei enää pysty kasvattamaan yksittäisten mikroprosessorien tehoa. Kellotaajuuden kasvu on pysähtynyt. Muiden mikroprosessorivalmistajien tilanne ei ole sen parempi.

Prosessoriteknologian kehitys ei silti ole pysähtynyt. Mikropiirille saadaan jatkossakin mahtumaan aina vaan enemmän transistoreja. Lisätransistoreilla ei kuitenkaan enää pystytä lisäämään yksittäisen mikroprosessorin tehoa.

Siksi Intel keskittyy tästedes valmistamaan *samalle mikropiirille monta mikroprosessoriydintä*.

Monesta mikroprosessorista saadaan moninkertainen teho vain jos ohjelma on luonteeltaan sellainen, että sen suorituksen saa hyvin pilkottua osiin. Kaksi remonttimiestä maalaa makuuhuoneen seinät hieman yli puolessa siitä ajasta mitä yksi — hieman yli puolessa, koska tarve väistellä toista aiheuttaa pientä ajanhukkaa. Mutta taideteoksen maalaamista ei voi nopeuttaa laittamalla kaksi taidemaalaria saman kankaan ääreen. Sotkuhan siitä vain tulisi.

Tietotekniikassa on monia makuuhuoneen seinien maalaamisen kaltaisia, helposti osiin pilkottavia tehtäviä. Sään ennustaminen lienee niistä tunnetuin. Siihen ja vastaaviin on iät ja ajat käytetty monesta prosessorista koostuvia supertietokoneita.

Sen sijaan kukaan ei tiedä, miten monen prosessorin teho saataisiin hyödynnettyä tavallisissa toimisto-ohjelmissa.

Jotakin pientä osataan toki tehdä. Yksi prosessori voi vastaanottaa saapuvia sähköpostiviestejä samalla kun toinen ajaa tekstinkäsittelyohjelmaa. Mutta tällä tavalla ei saada kovin montaa prosessoria työllistettyä, hyvä jos kaksi. Kyynikot tosin sanovat, että satakin prosessoria saadaan helposti kiireisiksi siten, että yksi ajaa käyttäjän ohjelmaa ja loput 99 koneeseen luvatta tunkeutuneita vakoilu-, virus- ja roskapostinlähetysohjelmia. Tämä ei kuitenkaan taida olla sellaista moniprosessoriteknologian hyödyntämistä, jota toivomme.

Intel haluaa, että ohjelmistotieteen tutkijat alkavat kehittää tekniikoita, joilla monen prosessorin tehosta saadaan hyöty irti joka kodin ja toimiston ohjelmissa.

Eivät ohjelmistotieteen tutkijat ole olleet sokeita tälle ongelma-alueelle tähänkään asti. Alan tutkimus alkoi käyttöjärjestelmien pulmista 1960-luvulla. Pikkuhiljaa se itsenäistyi omaksi alakseen nimeltä *rinnakkaisohjelmointi*, *concurrent programming*. Monia käytännöllisiä ratkaisuja on kehitetty. Samanaikaisesti teoreettisen tietojenkäsittelyn tutkijat ovat selvittäneet rinnakkaistamisen äärimmäisiä rajoja ja tutkineet rinnakkaisohjelmissa esiintyvien uudenlaisten ilmiöiden luonnetta.

Kuitenkin tähän asti rinnakkaisohjelmointi on ollut erikoisalue, josta useimpien ohjelmoijien ei ole tarvinnut välittää.

Mutta nyt tilanne muuttuu. Jos Intel saa tahtonsa läpi, niin *lähes kaikesta* ohjelmoinnista tulee rinnakkaisohjelmointia.

Ja jos Intel ei saa tahtoaan läpi, suorituskyvyn kasvu loppuu.

Asia ei tosin ole ihan näin yksinkertainen. Koska mikroprosessoritehoa on jo pitkään ollut valtavasti ja sitä on tähän asti tullut joka vuosi lisää ja lisää, ohjelmoijat ovat oppineet tuhlaamaan tehoa. Tämän päivän tietokone käynnistyy yhtä hitaasti kuin 1980-luvun tietokone, vaikka mikroprosessorit ovat nopeutuneet tuhatkertaisesti. Ei ole ollut mitään järkeä lisätä ohjelmointivaivaa pihistelemällä resurssia, jota on tarjolla yltäkylläisesti. (Tosin välillä tuntuu, että tuleva prosessoriteho tuhlataan jo ennakolta — niin hitaaksi PDF-dokumenttien avaaminen on muuttunut.)

Suorituskykyä saataisiin siis lisää tyytymällä nykyisiin prosessoreihin ja tekemällä ohjelmat paremmin. Tämä ei tietenkään ole se, mitä Intel haluaa. Intel on prosessorivalmistaja, ja se haluaa jatkossakin myydä uusia prosessoreita.

Siksi Intel haluaa, että tutkijat ratkaisevat rinnakkaisohjelmoinnin ongelmat ja ohjelmoijat siirtyvät sankoin joukoin rinnakkaisohjelmointiin.

Asia on Intelille strategisesti tärkeä. Ja Intelillä on rahaa.

Tästä pääsemmekin Intelin toiseen viestiin ETAPS-tapahtuman nyt jo viininhuuruiselle yleisölle. Intel, vaikka onkin mikroprosessorivalmistaja, palkkaa ohjelmistotutkijoita Englannin Cambridgen laboratorioonsa. Tarjous oli tehty hyvin houkuttelevaksi. Tutkimus on täysin julkista, mitään ei tarvitse piilottaa liikesalaisuuksien vuoksi, paikat soveltuvat erinomaisesti akateemista uraa luoville. Kaikkea on tarjolla kesätöistä pysyviin työsuhteisiin, lähettäkää ansioluettelonne tähän osoitteeseen. Viiniä on vielä jäljellä, ottakaa lisää.

Tuleeko Intel onnistumaan tavoitteessaan?

Siitä tuskin on pelkoa (tai toivoa), että ohjelmoijat päättäisivät valita rinnakkaisohjelmointiin siirtymisen sijasta entistä nopeampien ohjelmien tekemisen nykyisille prosessoreille. Harkitseva ja huolellinen työskentely ei päinvastaisista puheista huolimatta ole koskaan ollut ohjelmistoalan kuuma juttu, eikä tilanne ole viime vuosina muuttunut ainakaan parempaan suuntaan. Tarvittaisiin suurempia matemaattis-loogisia valmiuksia kuin valtaosalla ohjelmistoammattilaisista on, sekä paljon perusteellisempaa pilkunviilausta kuin kenenkään normaalin ihmisen mielenterveys kestää.

Sitäpaitsi ajatus uusien moniydinprosessorien vaatimaan “uuteen” huipputeknologiaan siirtymisestä on paljon helpompi markkinoida kuin ajatus työn tekemisestä entisellä teknologialla mutta nykyistä huolellisemmin. Jälkimmäisellä kun ei voi hypettää. Ehkä viiden vuoden kuluttua kohistaan jostakin mullistavasta ZYX-teknologiasta, jossa on valikoima rinnakkaisohjelmoinnin tutkijoiden iät ja ajat tuntemia asioita puettuna myyvään kuosiin.

Periaatteessa on olemassa kolmaskin vaihtoehto: se, että käyttäjät tyytyvät nykyisiin ohjelmiin. Viimeksi kuluneiden 20 vuoden aikana olen saanut loppujen lopuksi aika vähän uusia selvästi hyödyllisiä toimintoja käyttööni. Mutta se vähäkin on merkityksellistä. Vaikka pystyisin kirjoittamaan tekstini 20 vuoden takaisilla ohjelmilla melkein yhtä kätevästi kuin nykyisillä, en aivan yhtä kätevästi, sillä kuvien liittämiseksi tekstiin on nykyisin paljon enemmän mahdollisuuksia ja se on paljon vaivattomampaa kuin 20 vuotta sitten. Sitäpaitsi nykyisiin ohjelmiin tyytyminen on mahdoton ajatus myös siksi, että se ei sovi markkinoiden intresseihin.

Voidaan siis pitää melkoisen varmana, että lähivuosina tullaan kovalla voimalla yrittämään laajamittaista siirtymistä rinnakkaisohjelmointiin.

Nimenomaan yrittämään. Aiemman tutkimuksen viesti on, että rinnakkaisohjelmointi on vielä vaikeampaa kuin tavallinen ohjelmointi. Se on ollut käytännön kokemus. Lisäksi on olemassa vakavasti otettavia, joskaan ei missään nimessä sitovia kompleksisuusteoreettisia tuloksia, joilla on ikävä viesti. Niiden mukaan laajamittainen rinnakkaistaminen tehtävissä, joissa siitä olisi kovasti hyötyä mutta joissa se ei vielä ole onnistunut, saattaa olla periaatteessa mahdotonta.

Rinnakkaisohjelmoinnin varaan rakennetun toivon pettämisestä on ennakkotapaus. Vuoden 1990 tienoilla japanilaiset saivat eurooppalaiset hetkeksi huolestumaan ilmoittamalla kehittävänsä supertietokoneisiin ja tekoälystä ja logiikkaohjelmoinnista tuttuun Prolog-kieleen perustuvaa uudenlaista tietotekniikkaa. Tavoitteena oli päästää Prologin äly valloilleen rinnakkaisten tietokoneiden suuren laskentatehon avulla. Uuden tekniikan piti tehdä ohjelmoinnista helppoa, mullistaa tietojenkäsittelyn maailma ja tehdä Japanista alan johtava maa. Logiikkaohjelmointi ei kuitenkaan antautunut rinnakkaiseen suoritukseen perustuville supertietokoneille, ja hanke hiipui vähin äänin.

Rinnakkaisuus on nykyisissä jokatytön ja -pojan ohjelmointityökaluissa alkeellisella tasolla. Jopa aivan perusprimitiiveistä vallitsee sekaannusta, eikä primitiivejä ole yleensä sijoitettu kieliin vaan kirjastoihin. (Ada-kieleen kuuluva kohtaamismekanismi on tärkeä poikkeus.) Intelin tutkijat ja muut pystyvät varmasti viemään alaa lähivuosina eteenpäin.

Ehkä tässä tulee toistumaan se, mikä tapahtui dynaamiselle muistille ja osoittimien käytölle. 1980-luvulle tultaessa niitä pidettiin, jos ei nyt varsinaisesti vain huippuasiantuntijoille varattuna erikoisalueena, niin sen verran vaikeana kuitenkin, että niitä ei edes yritetty opettaa kaikille tuleville ohjelmoijille. Niiden käyttö olisi kyllä ollut perusteltua, sillä monet tietorakenteet tarvitsevat niitä. Varsinkin osoitin funktioon oli mysteeri. Eksoottisuudestaan huolimatta se oli käyttökelpoinen, sillä se tuo sellaista joustoa ja epähomogeeniseen tilanteeseen mukautumiskykyä, jota oli tuolloin vaikea muuten saavuttaa.

Sitten joku löysi toistakymmentä vuotta vanhassa, diskreettien tapahtumien simulointiin tarkoitetussa kielessä olleita käsitteitä, joiden avulla dynaamisen muistin käyttö, osoittimet ja osoittimet funktioihin voitiin pukea olennaisesti helpommin ymmärrettävään muotoon. Yhtäkkiä kaikki puhuivat niistä. Olio-ohjelmoinnin maailmanvalloitus oli alkanut. Nykyisin olioita, virtuaalisia funktioita ynnä muuta sellaista opetetaan kaikille.

Jos sama tapahtuu rinnakkaisohjelmoinnille, niin yhtenä peruskäsitteenä tulee kenties olemaan tilakone. Se on sellaisenaan tai naamioituna keskeisessä roolissa sekä valtaosassa rinnakkaisuuden teorioita että monissa käytännön mallinnusmenetelmissä.

Muilta osin rinnakkaisuuden sekä teoria että käytäntö ovat niin hajanaisia, että on todella vaikeaa arvata, mitkä käsitteet tulevat nousemaan muiden ylitse. Itse uskon, että Milnerin, Hoaren ja muiden luoma ulkoisen käyttäytymisen tai täyden abstrahoinnin (full abstraction) teoria on samassa suhteessa rinnakkaisohjelmointiin kuin Turingin koneiden teoria on klassiseen ohjelmointiin, mutta tällainen näkemys ei näytä olevan saamassa yleistä hyväksyntää. Ja vaikka saisikin, rinnakkaisohjelmoinnin perusteoriaa koskevasta yksimielisyydestä on vielä hyvin pitkä matka kuljettavana yleisesti hyväksyt-

tyihin käytännön kieliin ja menetelmiin. Sitäpaitsi käytäntö ei välttämättä nojaudu perusteoriaan kehityksessään.

Olisi siis uhkarohkeaa ennustaa, pystyykö lähivuosien tutkimus tuomaan rinnakkaisohjelmoinnin suurten ohjelmoijajoukkojen ulottuville vai onko se pohjimmiltaan liian vaikeaa. Edellinen vaihtoehto saattaisi johtaa laajaan ohjelmoijien täydennyskoulutustarpeeseen, aivan kuten olio-ohjelmoinnin tulo johti aikaisemmin. Jälkimmäinen vaihtoehto johtaisi koti- ja toimistotietokoneiden (hyödynnettävissä olevan) suoritustehon kasvun päättymiseen, mikä aiheuttaisi toisenlaisen mullistuksen alan teollisuudessa.

Viiden vuoden päästä tiedämme paljon enemmän kuin tänään.

— Antti Valmari —