Voorkom onzichtbare verspilling: energie-behoefte van software

Energie besparen dankzij zuinigere software

De afgelopen jaren is de energie-efficiëntie van ICT-systemen al sterk toegenomen als gevolg van de behoefte aan een langere levensduur van accu's in mobiele telefoons en laptops en de wens om de energiekosten in datacentra te verlagen. De meeste inspanningen zijn gericht op het verhogen van de efficiëntie van hardware en de daarvoor benodigde infrastructuur in het datacentrum. De vraag is echter of al deze hardware wel nodig is en of niet ook meer naar optimalisatie van applicaties gekeken moet worden. Zolang de software inefficiënt gebruikmaakt van de hardware, blijft het dweilen met de kraan open.

Jeroen Arnoldus en Joost Visser

Strikt genomen verbruikt software geen energie, want software is op zichzelf niks meer dan een heel gedetailleerde bouwtekening van een machine. Tijdens de interpretatie van de software door de computer wordt deze machine realiteit. Zo'n machine gebruikt echter wel energie en het zou – zoals gebruikelijk in andere disciplines – bekend moeten zijn of deze machine haar taken efficiënt uitvoert. Voor een mechanische machine, of een elektrisch apparaat, is dit redelijk eenvoudig te bepalen. Door de hoeveelheid nuttige uitgaande energie te delen door de ingaande energie wordt de efficiëntie berekend. Denk bijvoorbeeld aan een auto of een verwarmingsketel. Ook is meestal goed zichtbaar waar de meeste energie wordt omgezet; dit is te herkennen aan de componenten die warm worden. Helaas is dit bij computerapplicaties niet mogelijk. De uitvoer van de applicatie is geen energie, maar een statisch resultaat van een berekening. We kunnen lastig van de buitenkant beoordelen of de berekening efficiënt is uitgevoerd, aangezien dit in de meeste grote applicaties verborgen zit in een woud van processen en algoritmes.

Toch vallen er juist grote slagen te maken met het zuiniger maken van applicaties. Een efficiëntere applicatie heeft direct gevolgen voor de gehele energieketen, want er is minder hardware nodig, minder koeling, minder energietransport en er hoeft minder energie opgewekt te worden. In dit artikel gaan we in op een aanpak om applicaties te beoordelen op hun efficiëntie, en welke stappen er gezet moeten worden om uiteindelijk tot echt zuinige software te komen.

Samenvatting

Efficiëntere software heeft direct gevolgen voor de gehele energieketen, want er is minder hardware nodig, minder koeling, minder energietransport en minder energie. Er zijn diverse quick wins waarmee veranderingen om applicaties energiezuiniger te maken soms snel in te voeren zijn. Om tot energiezuinige software te komen moet dit een zorg zijn voor de gehele organisatie en softwarelevenscyclus.

De datafabriek

Zoals eerder geschetst, kan een stuk software vergeleken worden met een bouwtekening van een machine, heel specifiek een machine die data manipuleert. In essentie doet een applicatie niks anders dan data verplaatsen onder bepaalde condities. Deze condities maken het mogelijk om data te filteren, sorteren, veranderen et cetera. Grote applicaties bestaan meestal uit meerdere losse machines, ook wel processen genaamd. Losse processen maken het mogelijk taken parallel aan elkaar uit te voeren, verantwoordelijkheden te scheiden en het overzicht te vergroten. Elk proces voert zijn eigen specifieke taak uit en de processen onderling kunnen elkaar direct berichten versturen of zijn gekoppeld via een berichtentransportlaag (of wel bus). Zo zijn er processen die bedrijfsregels uitvoeren, data herverpakken in andere formaten of processen die batches uitvoeren op grote datasets.

Door een applicatie te beschouwen als een set van processen die met elkaar communiceren valt er een interessante analogie te maken met een productielijn van machines en transportbanden. De processen zijn de machines en de communicatie tussen de processen is de transportband. Bij productielijnen is het duidelijk welke machines er zijn, welke taken ze uitvoeren en welke hulpbronnen ze nodig hebben. Tijdens de productie valt ook goed te zien en te meten welke machines efficiënt werken, waar veel hulpbronnen nodig zijn en welke machines weinig uitvoeren. Dit inzicht ontbreekt helaas in veel softwareapplicaties, zeker bij applicaties die onderdeel zijn van een groter ICT-landschap.

Energiezuinige software

Net als bij een productielijn moet elke activiteit onder de motorkap van de applicatie waarde toevoegen. Is het omzetten van het ene XML-bericht in het andere XML-bericht wel noodzakelijk? Is het nodig om de machtsfunctie te gebruiken als elke keer een kwadraat wordt gebruikt? Is het efficiënter om het filter uit te voeren op de centrale database of op de decentrale personal computer? Is het wel nodig alle data persistent op te slaan? Deze vragen kunnen niet in het algemeen beantwoord worden, maar de architect en programmeurs zouden deze vragen wel continu aan zichzelf moeten stellen. Dat zij soms krampachtig vasthouden aan bepaalde architecturen en eerder genomen ontwerpbeslissingen helpt hierbij niet.

Een eerste stap richting energiezuinige software is het in kaart brengen van de onderliggende processen en hun relaties om tot een energiemodel van de applicatie te komen. Als voor een applicatie een model is gemaakt van de processen en hun relaties, kan er, net als bij een productielijn van een fabriek of een logistieke keten, geredeneerd worden over optimalisaties (Cortellessa, Di Marco & Inverardi, 2011). Aangezien we in het geval van groene software specifiek op energieverbruik willen optimaliseren, moeten de processen worden voorzien van informatie over hun opgenomen vermogen. Het energieverbruik van processen valt op te splitsen in het opgenomen vermogen in rust en het opgenomen vermogen per transactie. Daarnaast moeten de communicatielijnen tussen de processen worden voorzien van informatie over het energieverbruik per verstuurde eenheid van data.

In het ideale geval worden deze vormen van energieverbruik gemeten, maar helaas is dat in de praktijk niet altijd haalbaar, zoals wij hebben ervaren tijdens proefprojecten met bestaande systemen bij onze klanten. In dat geval moeten er aannames over opgenomen vermogen gemaakt worden. In de praktijk blijkt dat deze aannames, gebaseerd op eerder uitgevoerd onderzoek, goed werken.

Nadat het model van de processen en relaties is voorzien van de opgenomen vermogens, kan het gebruikt worden om de energie van de applicatie in rusttoestand te berekenen en het opgenomen vermogen per transactie te bepalen. Hiervoor dienen eerst de meest gebruikte functionaliteiten van de applicatie bekend te zijn. Voor elk van deze

functionaliteiten kan het opgenomen vermogen bepaald worden en kan worden berekend welke processen de meeste invloed hebben op de benodigde energie.
Als het consumptiemodel in kaart is gebracht, kan gericht geoptimaliseerd gaan worden. Een mogelijkheid hiervoor is het toepassen van het 'lean manufacturing'-principe (Krafcik, 1988). De processen moeten worden beoordeeld op toegevoegde waarde. Deze toegevoegde waarde kan worden bepaald voor de geleverde functionaliteit en de hoeveelheid energie nodig voor de uitvoer daarvan. Is er gebruikgemaakt van het meest geschikte algoritme? Worden energievretende subprocessen onnodig vaak aangeroepen? Dit vertoont een grote gelijkenis met het optimaliseren van productieprocessen en logistieke ketens. Het grote probleem is dat het heen en weer rijden met lege vrachtwagens voor iedereen zichtbaar is, maar bijvoorbeeld het nutteloos opnieuw overhevelen van JavaScript-bibliotheken niet.

Besparingen simpeler dan gedacht

Veranderingen om de applicatie energiezuiniger te maken zijn soms snel in te voeren. Met de volgende quick wins (Semeijn, 2011) kan gemakkelijk bespaard worden op energieverbruik:

- Reduceer het noodzakelijke dataverkeer door alleen het broodnodige over te sturen, bijvoorbeeld door de resolutie van afbeeldingen te verlagen.
- Sla niet alle historische gegevens op. Het is best voorstelbaar dat het soms handig is dat soort data nog te kunnen oproepen, maar het levert wel een enorme berg gegevens op die in de praktijk maar zelden worden ingekeken. Het opslaan en archiveren van al deze data kost opslagruimte, tijd en energie. Ook het doorzoeken van grote datasets gaat gepaard met de benodigde computerkracht. Tijdig filteren van deze data voorkomt wildgroei aan gegevens en de daarbij behorende opslagkosten en energieverbruik.
- Verlaag het opgenomen vermogen van de applicatie in rust. Dit kan bijvoorbeeld door deze op gedeelde servers uit te rollen. Het probleem van computers is dat deze machines in rust nog steeds een significante hoeveelheid energie verbruiken en dat computers optimaal presteren onder volle belasting. Door meerdere applicaties op gedeelde servers te laten draaien wordt de kans dat de computer een hoge belasting heeft groter, waardoor het rendement toeneemt. Zorg dat de applicaties zo opgezet zijn dat ze niet afhankelijk zijn van de onderliggende hardware, zodat ze gemakkelijk op gedeelde servers kunnen worden uitgerold.
- Protocollen tussen processen dienen zo opgezet te zijn dat de verschillende processen niet onnodig vaak contact maken. Een verbinding opzetten kost energie en tijd, zeker in het geval van mobiele communicatie. Door zoveel mogelijk data tegelijk te versturen wordt de efficiëntie van de dataoverdracht verhoogd.
- Maak in de productieomgeving gebruik van gecompileerde talen. Veel webapplicaties zijn geschreven in een programmeertaal die gebruikmaakt van een interpreter, die de programmatekst tijdens het gebruik van de applicatie naar machinecode vertaalt. Deze constructie kost de computer extra rekenkracht.
- Versimpel de infrastructuur door bijvoorbeeld het gebruik van proxy's als adapter tussen systemen te verwijderen. Deze proxy's hebben niet altijd toegevoegde waarde, zoals extra veiligheid of bescherming van de achterliggende server tegen piekbelastingen. Als er geen zichtbare toegevoegde waarde is, zou zo'n proxy geëlimineerd kunnen worden, wat kan resulteren in energiebesparingen.

Een energiebewuste softwareorganisatie

Om tot energiezuinige software te komen moet dit een zorg zijn voor de gehele organisatie en softwarelevenscyclus. Helaas is dit soms lastig te bereiken als er een te grote scheiding van verantwoordelijkheden is tussen het ontwikkelproject en het technisch beheer. Het ontwikkelteam is meestal gericht op het implementeren van de functionaliteit binnen de gestelde tijd en budget, terwijl het technisch beheer vooral de applicatie operationeel moet houden voor het benodigde aantal gebruikers. Om aan te geven op welk volwassenheidsniveau van energiebewustzijn met betrekking tot software de organisatie zich bevindt, hanteren we een model van vijf niveaus (Keijzer, 2010) zoals in figuur 1.

volwassenheids-niveau	energiebewustzijn m.b.t. software
5	energiebesparing onderdeel van hele softwarelevenscyclus, met formele communicatie tussen beheer en ontwikkelteams
4	verschillende afdelingen overleggen met elkaar pijnpunten m.b.t. energiebesparing
3	energiebesparing op de agenda van de hele organisatie
2	bedrijfsleiding geeft ruimte om verspilling te verkleinen
1	ad hoc besparen van energie

Figuur 1. **Volwassenheidsniveaus van organisaties met betrekking tot energiebesparing**

»Om een applicatie te optimaliseren is betere samenwerking tussen beheerders en ontwikkelaars vereist«

Niveau 1 is ad hoc. Dit houdt in dat de organisatie zich realiseert dat een applicatie een bepaalde hoeveelheid energie verbruikt. De organisatie zal passende maatregelen nemen om de benodigde energie terug te dringen. Denk hierbij aan het uitzetten van applicaties buiten kantooruren.
Niveau 2 is dat van commitment. Dit is een situatie waarin de bedrijfsleiding zich realiseert dat applicaties ook energie verspillen. De leiding geeft ruimte om de verspilling te verkleinen door mensen te informeren en energiebesparende plannen te bedenken.
Niveau 3 betekent dat energiebesparing op de agenda staat van de gehele organisatie. Dit betekent dat zowel de ontwikkelteams, het technisch beheer als de gebruikers zich bewust zijn van het energieverbruik van applicaties.
Niveau 4 heeft als sleutelwoord 'integratie', te weten integratie tussen de ontwikkelaars, de gebruikers en vooral de beheerders van de applicatie. Deze groepen weten van elkaar waar eventuele pijnpunten liggen. De beheerders kunnen het energieverbruik meten en dat voorleggen aan de ontwikkelaars. De gebruikers kunnen aangeven welke functionaliteiten belangrijk zijn en welke een hoge beschikbaarheid vereisen. Dit overleg helpt om de applicatie gericht te optimaliseren voor een lager energieverbruik.
Niveau 5 is het hoogste niveau en is tevens het ideaalbeeld. Bij dit niveau is energiebewustzijn een onderdeel van alle stadia van de softwarelevenscyclus. Er is een vastgelegde terugkoppeling naar alle stadia, zodat verspilling gemeten in de productieomgeving gerapporteerd wordt aan de ontwikkelafdeling. Ook wordt technisch beheer direct betrokken bij het architectuurontwerp om aanwijzingen te geven over mogelijke energieconsequenties van de genomen beslissingen. Op deze manier wordt de energieconsumptie continu geoptimaliseerd tot het minimale niveau en wordt het rendement van de applicatie verhoogd.

De wet van Wirth

Het afgelopen decennium zijn performance en energieverbruik geen hoofdprioriteit geweest tijdens het ontwikkelen van applicaties. De hardware werd steeds sneller en kon meer instructies uitvoeren per watt (zie figuur 2). De wet van Wirth (1995) stelt echter:

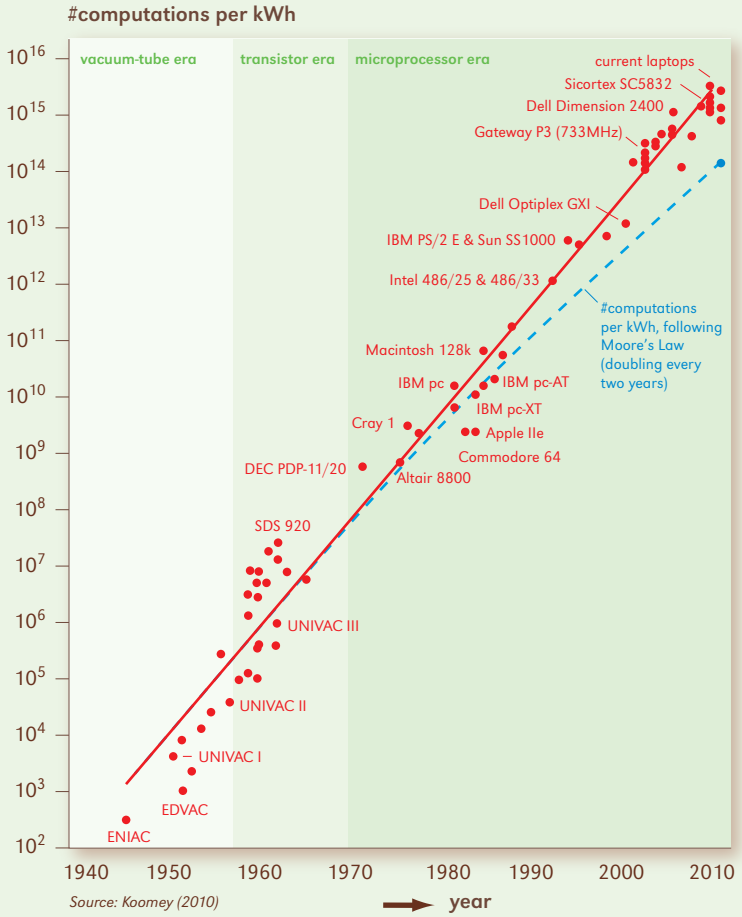
> 'Software wordt sneller trager dan hardware sneller.'

Dit merken we bijvoorbeeld bij de aanschaf van een nieuwe laptop. Die is sneller. Maar er staan ook nieuwere, zwaardere softwareversies op. Een spreadsheet invullen, een document schrijven, een presentatie maken – het is er niet sneller op geworden.
De vraag is in hoeverre de wet van Wirth ook opgaat voor de efficiëntie van de hardware. Processors worden steeds zuiniger – per jaar meer dan twee keer zo zuinig (Koomey, 2010) – maar deze vooruitgang moet niet worden tegengewerkt door inefficiëntere applicaties. Voordat we met meer zekerheid antwoord op deze vraag kunnen geven, moeten we beter begrijpen wat de impact van het uitvoeren van software is op het energieverbruik van computers.
Energieconsumptie in computers is geen triviaal onderwerp. Uiteindelijk zijn niet alleen de uitgevoerde instructies van belang, maar ook de verwerking in de processor, de soort data, de hoeveelheid data, in welk geheugen de data zijn opgeslagen en de noodzakelijke perifere hardware. De energieconsumptie van een stuk software kan daardoor enkel bepaald worden door werkelijk aan de hardware te meten. Onderzoek is hiervoor noodzakelijk, zoals onder andere wordt uitgevoerd door het recent opgerichte Software Energy Footprint Lab (SEFLab). Het SEFLab is een samenwerkingsverband tussen het CleanTech-onderzoeksprogramma van de Hogeschool van Amsterdam (HvA) en de Software Improvement Group (SIG).

Afsluitend

Het ontwikkelen van kennis rond het energiezuiniger maken van applicaties is in volle gang. De grootste winsten worden op dit moment geboekt bij het optimaliseren van hardware en datacentra. Er kunnen echter ook veel besparingen behaald worden door de applicaties zelf te optimaliseren. Binnen het door AgentschapNL ondersteunde kennisnetwerk Green Software (KNGS) wordt door partijen kennis over energiezuinige software verzameld en onderzoek gestimuleerd. Het doel is efficiënte hardware nog efficiënter te gebruiken. Dit helpt niet alleen de kosten voor energieverbruik te verlagen, maar, door reductie van het benodigde aantal machines, ook de kosten van beheer en afschrijving van de infrastructuur.

Voordat deze optimalisaties kunnen worden gerealiseerd, moet eerst het werkelijke energieverbruik van softwareapplicaties worden bepaald. Deze meetresultaten van het energieverbruik in test- of productieomgevingen moeten worden geprojecteerd op de processen binnen de applicatie. Het ontwikkelteam kan deze gegevens gebruiken om de applicatie te optimaliseren. Dit vereist een betere samenwerking tussen beheerders en ontwikkelaars. Om deze samenwerking te stimuleren zijn energieverbruikseisen als onderdeel van de kwaliteitseisen voor de applicatie noodzakelijk. Hierdoor wordt energieverbruik een onderdeel van het ontwerpproces, zodat in een vroeg stadium een afweging tegen andere eisen gemaakt kan worden.



Figuur 2. **Ontwikkeling van processorsnelheid versus energieverbruik** (Koomey, 2010)

Literatuur

Cortellessa, V., A. Di Marco & P. Inverardi (2011). *Model-Based Software Performance Analysis*. Springer.

Keijzer, R. (2010). Hoeveel energie verbruikt de software eigenlijk? *Automatisering Gids*, 17 december 2010.

Koomey, J.G. (2010). Outperforming Moore's Law. *IEEE Spectrum*, March, http://spectrum.ieee.org/computing/hardware/outperforming-moores-law.

Krafcik, J. (1988). Triumph of the lean production system. *Sloan Management Review* 30 (1), pp. 41-52.

Semeijn, B. (2011). 8 tips om software energiezuinig te maken. *Computerworld*, 17 november 2011, http://computerworld.nl/article/13333/8-tips-om-software-energiezuinig-te-maken.html.

Wirth, N. (1995). A Plea for Lean Software. *Computer* 28 (2), pp. 64-68.

Dr. Joost Visser
is hoofd research bij de Software Improvement Group (SIG) en voorzitter van het KNGS. E-mail j.visser@sig.eu.

Dr. Jeroen Arnoldus
is consultant bij de Software Improvement Group (SIG). E-mail j.arnoldus@sig.eu.