

Compiler Design
Prof. Y. N. Srikant
Department of Computer Science and Automation
Indian Institute of Science, Bangalore

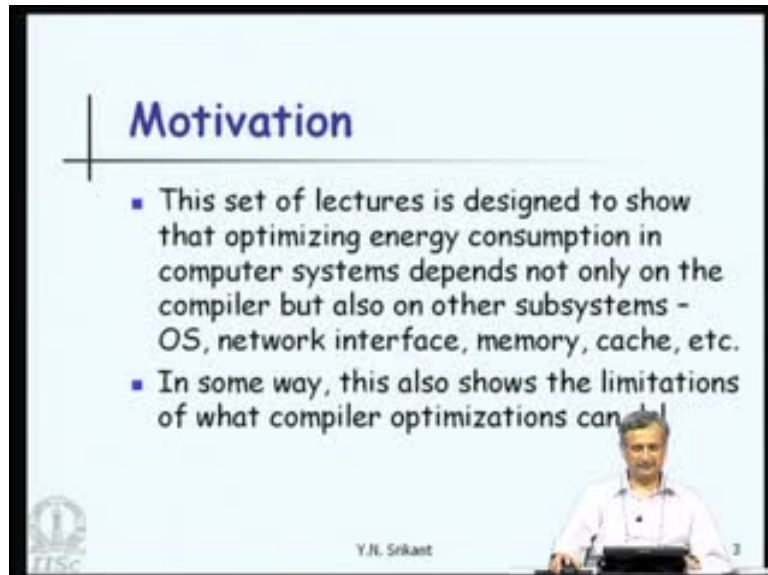
Module No. # 17

Lecture No. # 32

Energy-Aware Software Systems

Welcome to the lecture on energy-aware software systems. It is a digression from the main topic of compilation, but there is a reason for it. So let me state the motivation first.

(Refer Slide Time: 00:30)



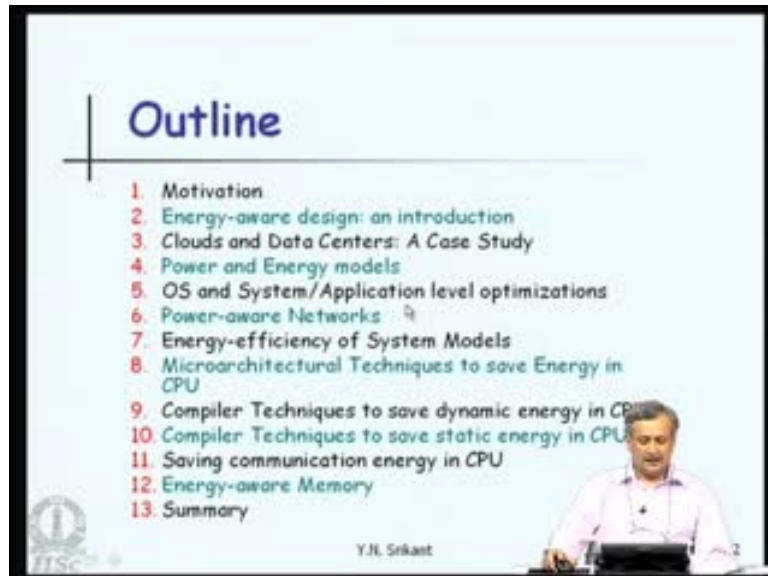
Then, look at the contents of the lecture. Suppose, we want to optimize energy consumption in a computer system, it is not enough if we optimize the program for the sake of energy consumption.

The energy consumption in a complete computer system actually depends on almost every other subsystem – the operating system, the network interface, the memory, the cache and any other peripherals attached to it, like disk, and so on, so forth.

If you simply optimize the program and reduce the energy consumption of CPU, the overall energy in the system may not be minimized. In fact, the CPU energy may be only about 25 percent of the total energy consumption. So, in some sense, this also shows

what a compiler optimization cannot really do. Hence, there is much more to read than just compiler optimization.

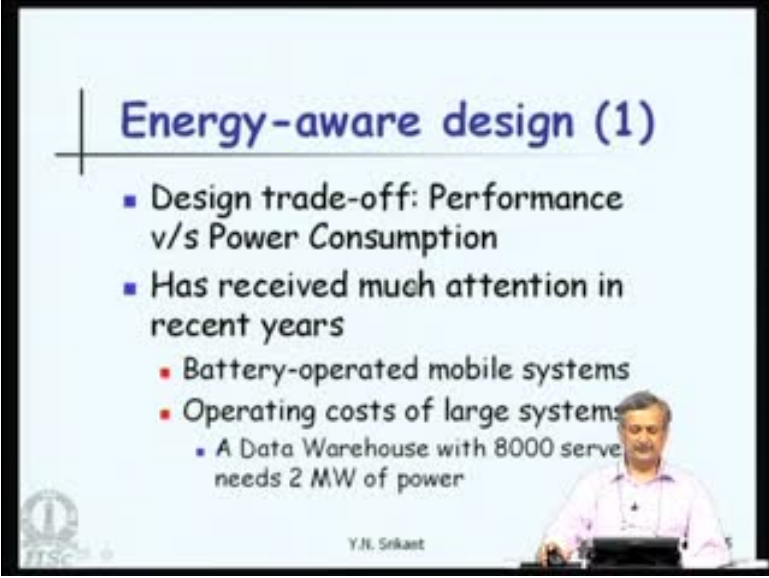
(Refer Slide Time: 02:05)



To get a feel of such a limitation, it will be necessary for us to look at other things as well. Let me go back one step. We are going to look at the reasons – why energy-aware design is important in the introduction. Then, we will look at a case study on clouds and data centers to see what is the magnitude of saving that we can get and how. We also need to look at models for power and energy consumption because the basis for any optimization, either in a compiler or operating system, depends on how good the model is.

We will also consider operating system- and system application-level optimizations, power-aware networks, energy efficiency of system models, micro-architectural techniques to save energy in CPU, compiler techniques to save dynamic and static energy in CPU, saving communication energy in CPU, energy-aware memory, and of course, summary.

(Refer Slide Time: 03:17)



Energy-aware design (1)

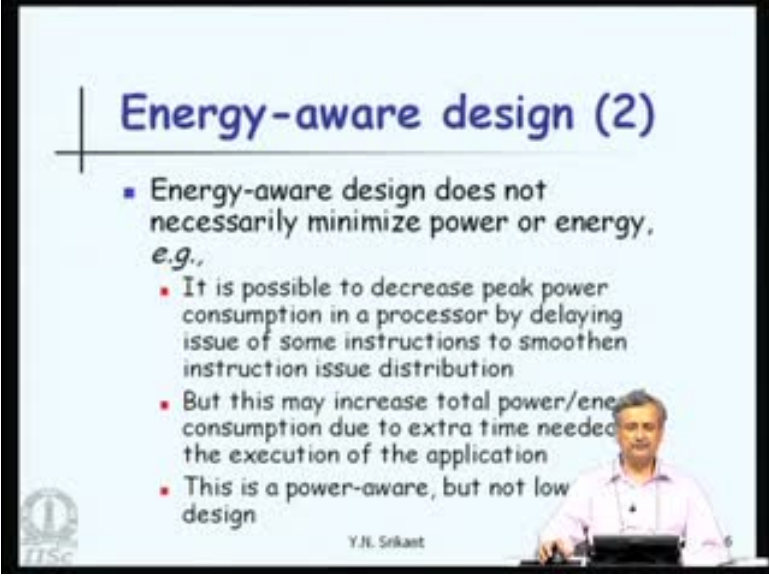
- Design trade-off: Performance v/s Power Consumption
- Has received much attention in recent years
 - Battery-operated mobile systems
 - Operating costs of large systems
 - A Data Warehouse with 8000 servers needs 2 MW of power

Y.N. Srikant

Let us see what exactly is energy-aware design? Any software design will be a trade-off. Do we want to increase the performance of the software or do we want to reduce the power consumption? If we increase the speed of CPU, the performance will go up, but power consumption will also go up. And if you reduce the power consumption, by say, lower than the voltage, the performance will also go down. So, it is very difficult to increase performance and reduce power consumption at the same time. But that is our goal. We want to provide the same level of performance for software, but we would like to consume much less power than we used to.

Power consumption reduction, energy reduction, etc., have received a lot of attention in the last few years. For example, we have battery-operated mobile systems; we want to retain the charge on the battery as long as possible. That is at the lower end. At the higher end, if you look at a data center, a large one, say, with 8000 servers, the power consumption would be 2 megawatts. Any reduction in this power will actually translate to money directly. Energy-aware design does not necessarily minimize power or energy.

(Refer Slide Time: 04:54)



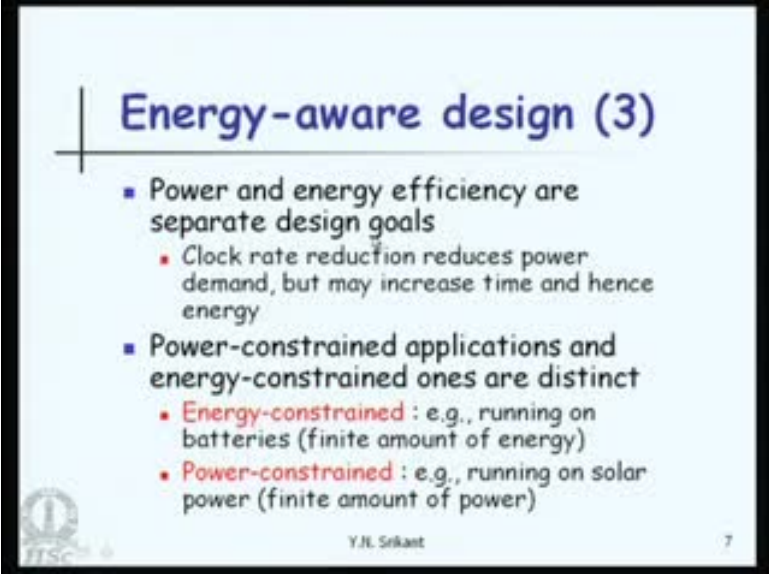
Energy-aware design (2)

- Energy-aware design does not necessarily minimize power or energy, *e.g.*
 - It is possible to decrease peak power consumption in a processor by delaying issue of some instructions to smoothen instruction issue distribution
 - But this may increase total power/energy consumption due to extra time needed for the execution of the application
 - This is a power-aware, but not low power design

Y.N. Srikant

Let us see what it really means. For example, it is possible to decrease peak power consumption in a processor by delaying issue of some instructions to smoothen instruction issue distribution. This is a fairly well-known thing. Peak power consumption will come down but this will increase the total power energy consumption due to the extra time needed to execute the application, because we have delayed instructions. More time will be needed to execute the application. That means, the chip is on for a longer duration, resulting in more power consumption. This is a power-aware design. In other words, we have reduced the power consumption – peak power consumption. But, it is not a low-power design because it does not reduce the power consumption.

(Refer Slide Time: 06:00)



The slide is titled "Energy-aware design (3)" and contains the following content:

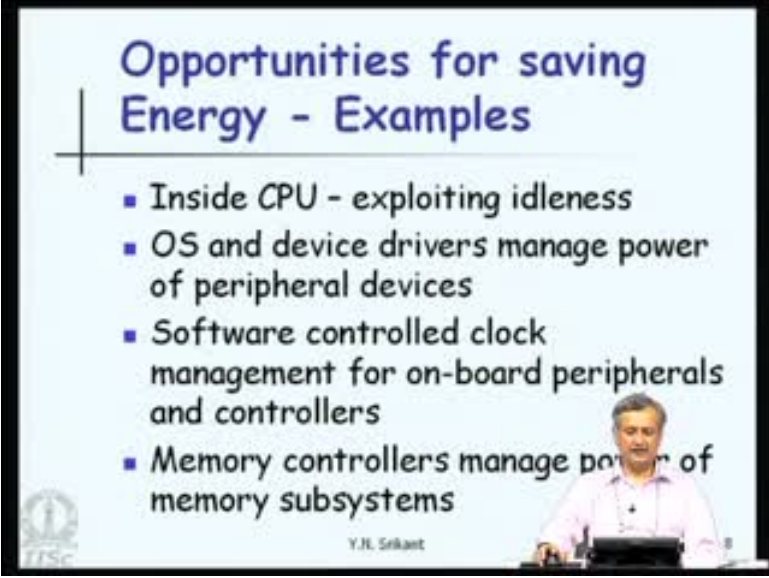
- Power and energy efficiency are separate design goals
 - Clock rate reduction reduces power demand, but may increase time and hence energy
- Power-constrained applications and energy-constrained ones are distinct
 - **Energy-constrained** : e.g., running on batteries (finite amount of energy)
 - **Power-constrained** : e.g., running on solar power (finite amount of power)

At the bottom left of the slide is the IITSC logo, and at the bottom center is the name "Y.N. Srikant". The number "7" is in the bottom right corner.

Power and energy efficiency are separate design goals. For example, clock rate reduction reduces the power demand but it may increase time and hence energy consumption. Why? Chip clock rate will reduce the power consumption, but the frequency being low, you need more time to execute the program.

Power constrained applications and energy constrained applications are also distinct. We are actually clarifying terminology here. Energy constrained applications are, for example, running on batteries. There is a finite amount of energy; we need to make the best use of it; so we have energy constrained applications running on it. Power constrained application, for example, running on solar power – there is a finite amount of power available because it depends on the size of solar cells. You really cannot increase it without increasing the size of solar array.

(Refer Slide Time: 07:13)



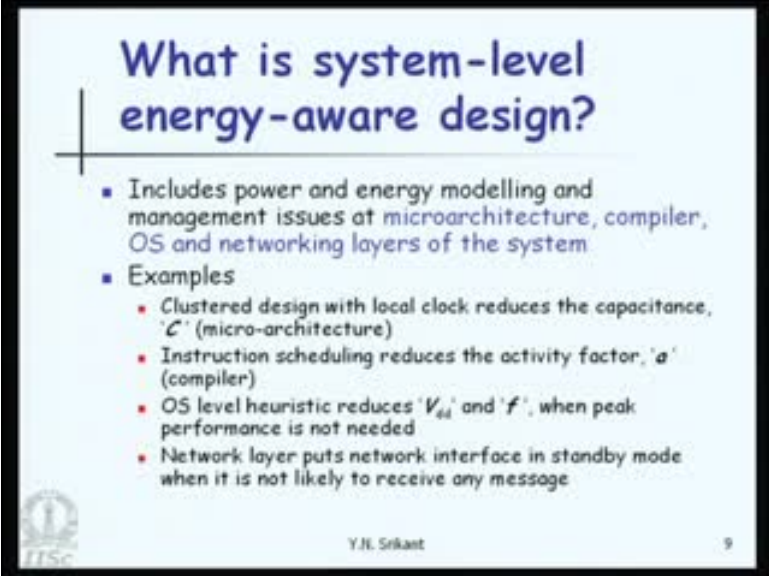
Opportunities for saving Energy - Examples

- Inside CPU - exploiting idleness
- OS and device drivers manage power of peripheral devices
- Software controlled clock management for on-board peripherals and controllers
- Memory controllers manage power of memory subsystems

Y.N. Srikant

To give you some examples of how to save energy inside the CPU, we can exploit the idleness of CPU. In other words, whenever the CPU is idle, we could put in a small piece of hardware inside, which switches off parts of the CPU, say may be, the ALU's, cache, and so on. Similarly, the operating system and device drivers can manage the power of peripheral devices very intelligently when the device is not being used – perhaps the power that is supplied to it can be cut off, or the voltage can be reduced, etc. This is seen very readily in the display of laptops and mobile phones. The display switches off after a few seconds in a mobile or after 10 to 15 minutes in a laptop, so that if the user is not using the laptop or mobile, there is no need to waste power on the display. Software controls clock management for on-board peripherals, so that the clocks speed could be reduced or increased as per requirement. Memory controllers manage power of memory sub-systems. It is possible to place memory blocks in low power mode when the memory is not being used at all, for a fair amount of time, and bring it back to life when there is a demand for memory. These are a few examples of how to save energy and power inside or outside a CPU. We will see more of this as we go on.

(Refer Slide Time: 09:23)



What is system-level energy-aware design?

- Includes power and energy modelling and management issues at microarchitecture, compiler, OS and networking layers of the system
- Examples
 - Clustered design with local clock reduces the capacitance, ' C ' (micro-architecture)
 - Instruction scheduling reduces the activity factor, ' a ' (compiler)
 - OS level heuristic reduces ' V_{dd} ' and ' f ', when peak performance is not needed
 - Network layer puts network interface in standby mode when it is not likely to receive any message

ITSC Y.N. Srikant 9

What is system-level energy-aware design? Such a design includes power and energy modeling. It also includes management issues at micro-architectural-level, compiler-level, OS-level, and also at networking layers of the system. We are looking at optimization at all levels. Let us see what these means, with some examples.

Consider a cluster design with local clock. In other words, there are clusters of function units, not just one function unit in the CPU, and instead of having a global clock for the several clusters, there would be a local clock for each cluster. This is a well-known design strategy. It reduces capacitance of the entire system. This is micro-architectural-level design consideration, which can reduce power requirements because once the capacitance decreases, the charge and discharge, etc., will be less, and power consumption will be reduced.

Instruction scheduling reduces activity factor a and this is a compiler-level optimization. If a function unit is not used, for example, for a few cycles, based on the history of usage, the architecture would have switched off the function unit. When an instruction executes again on this particular unit, it is brought back to life.

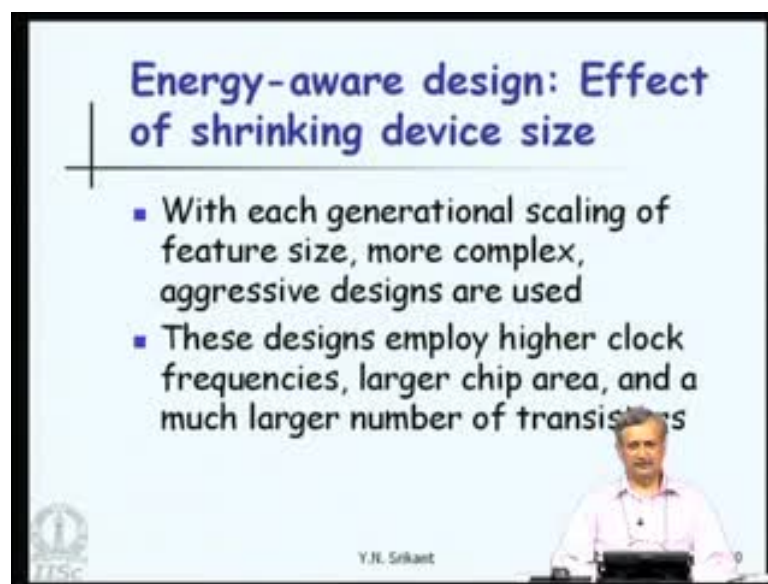
If this happens at random, there may be wastage of power because switching on and switching off of the function unit requires much more power than executing files – executing something or staying idle. What an instruction scheduler a compiler can do is to rearrange instructions, so that a function unit, which is busy, keeps on doing work, that

is, it keeps on executing instructions. And a function unit, which is idle remains idle as long as possible.

In other words, by rearranging instructions, we try to increase the busy periods and also the idle periods of function units. So, we reduce the number of transitions from high to low and low to high, thereby saving energy and power. Operating system-level heuristics reduce the source voltage V_{dd} and the frequency, when peak performances not needed.

If there are programs running on the system and you know the operating system priority, knows that a particular program can run slowly; there is no hurry for it to complete, when that particular program is running. The operating system can reduce the source voltage and the frequency of the chip, and thereby reducing the energy consumption.

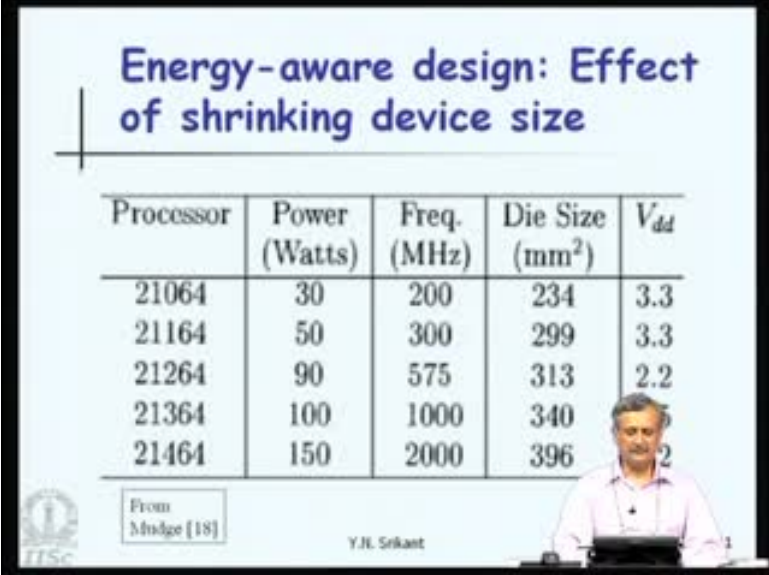
(Refer Slide Time: 13:20)



The network layer puts network interface in standby mode, when it is not likely to receive any message. Thereby, there is an obvious saving of power. Let us consider the effect of shrinking device size on energy-aware design. Why? Why should we do that? The point is with each generational scaling of feature and size, more complex and aggressive designs are being used. We all know that when integrated circuits were introduced in 70s, they had only a few gates on them. Slowly gradually, now, they have billions of gates and transistors on them.

This is possible only because there is a reduction in size of each one of these devices. So, there is more area available on the chip; hence, more complex and aggressive designs are now being used. These designs employ higher clock frequencies, they use larger chip area and much larger number of transistors.

(Refer Slide Time: 14:25)



Energy-aware design: Effect of shrinking device size

Processor	Power (Watts)	Freq. (MHz)	Die Size (mm ²)	V _{dd}
21064	30	200	234	3.3
21164	50	300	299	3.3
21264	90	575	313	2.2
21364	100	1000	340	1.5
21464	150	2000	396	1.2

From Mudge [18]
Y.N. Srikant

For the alpha processor and generations of them, let us see how is the power utilization. For 21064, the frequency was 200 megahertz, die size was 234 millimeter square, power consumption was 30 watts and power supply was 3.3 volts. As the next generation devices were introduced, the size of the die reduced, and it actually increased the supply. For example, from 234 it became 299, 313, 340, 396, etc. So the devices became more and more complex, but the frequency also increases – 300, 575, 1000, 2000, etc. The supply voltage really reduced – 3.3, 2.2, 1.5.; 21464 was 1.2 volts, die size is 396, frequency is 2000 megahertz or 2 gigahertz, but the power consumption is 150 watt.

(Refer Slide Time: 15:57)

Energy-aware design: Effect of shrinking device size

- The result is a significant increase in power dissipation
- Conclusion: Shrinking device size does not imply less power dissipation

Y.N. Srikanth

The slide features a presenter, Y.N. Srikanth, at the bottom right. The background is light blue with a white border. The title is in a large, bold, blue font. The bullet points are in a smaller, bold, blue font, with the first one highlighted in red. The presenter is a man with short grey hair, wearing a light pink shirt, standing behind a podium.

Power consumption, even though the device size has reduced, has increased because the number of devices has really gone up by leaps and bounds. The size of the chip and the number of devices being large, results in higher participation. That means, shrinking device size does not imply less power dissipation, as one may think.

(Refer Slide Time: 16:13)

Energy-aware design: Effect of power density on chips

Watt/cm²

From Unsal et al. [4]

Power density of Intel chips

Y.N. Srikanth

The slide features a presenter, Y.N. Srikanth, at the bottom right. The background is light blue with a white border. The title is in a large, bold, blue font. Below the title is a bar chart showing the power density of Intel chips. The y-axis is labeled 'Watt/cm²' and ranges from 0 to 1000. The x-axis lists Intel chip models: 386, 486, Pentium, PentiumPro, PentiumIII, PentiumIV, and Pentium. The bars show an increasing trend in power density. The presenter is a man with short grey hair, wearing a light pink shirt, standing behind a podium.

Chip Model	Power Density (Watt/cm ²)
386	~10
486	~20
Pentium	~80
PentiumPro	~100
PentiumIII	~300
PentiumIV	~400
Pentium	~600

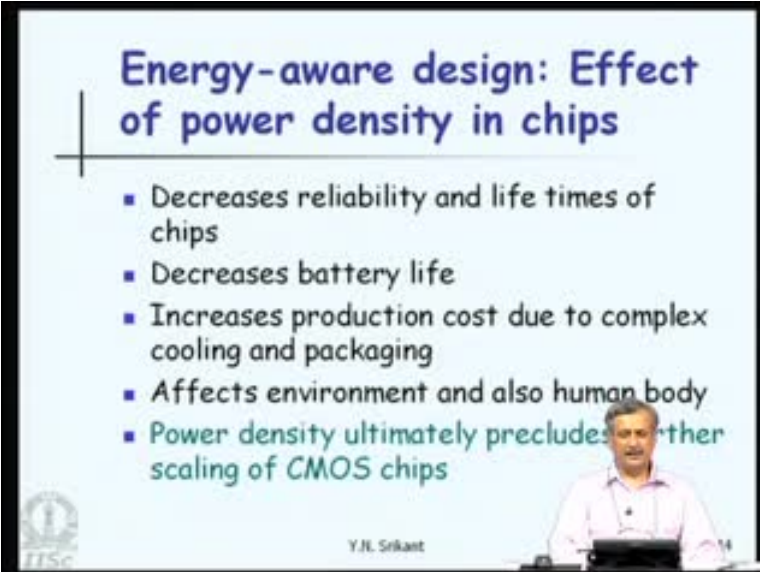
Let us also look at the power density on chips. This is a slightly old graph, but it suffices to give us the information that we require. Starting from 386 here, as we went to 486,

pentium, pentium pro, and to pentium 4, there is a steady increase in the power density. Watt per centimeter square is on this side, so there has been a steady increase.

Pentium 4 is at some level and you can see that; the next bar corresponds to a nuclear reactor. In other words, pentium 4 is only slightly lower at power density, or it is only slightly less hard compared to the inside of a power nuclear reactor. After the nuclear reactor, there would be the sun.

At Pentium 4 level, there is already probably the maximum or just below the maximum, as far as we have reached that maximum, as far as the power density is concerned. This is the reason why multi-core chips have been introduced. If we look at the multi-core chip specification carefully, we will see that the frequency of each core is lower and the number of cores per processor is more than one. What really has happened is you have reduced the power density, and frequency being lower, the power consumption will also be lower.

(Refer Slide Time: 18:26)



The slide features a title 'Energy-aware design: Effect of power density in chips' in blue text. Below the title is a list of five bullet points, each preceded by a blue square. The text of the slide is as follows:

- Decreases reliability and life times of chips
- Decreases battery life
- Increases production cost due to complex cooling and packaging
- Affects environment and also human body
- Power density ultimately precludes further scaling of CMOS chips

In the bottom right corner of the slide, there is a small inset image of a man in a light blue shirt sitting at a desk with a laptop. The name 'Y.N. Srikant' is printed below the inset image. In the bottom left corner, there is a small circular logo with the letters 'ITSC' inside.

Power density on the chips has reduced, but to make up for the loss of performance, the number of cores has been increased, so that if the software programmers are smart enough, they actually give you the same performance. But, increasing the power density in chips decreases reliability and lifetimes of chips – it decreases battery life. It increases power production cost due to complex cooling and packaging. It effects the environment

and the human body. Power density ultimately precludes further scaling of CMOS chips. I have already mentioned this.

(Refer Slide Time: 18:50)

Classification		Techniques	
Software	Application	Algorithm Design	Scheduling
	Virtual Machine	Resource Hibernation	Memory Management
	Compiler	Fidelity Reduction	Energy Accounting
	Operating System	Remote Execution	
Hardware	Architecture	Clock, Memory, and Interconnect Optimizations	
	Gates	Technology Mapping	Gate Restructuring
	Transistors	Transistor Sizing	Transistor Ordering

Leakage Reduction

Y.N. Srikant

What are the various techniques available at various levels to achieve energy savings? At the lowest level, we have transistors, gates, architecture. The software layer is above – operating systems, compiler, virtual machine applications. At various levels, let us take a look at the techniques, which are used at the transistor-level. The size of the transistors and transistor ordering controls the energy consumption at the gate-level technology mapping. Which particular technology at the gate restructuring control energy consumption?

At the architecture-level, clock, memory and interconnect optimizations, which we are going to see later, will control power consumption and energy consumption. With applications, virtual machines, compilers and operating systems, we have a host of techniques, which interact with each other. That is the reason why there is no separation between these techniques.

We can do better algorithm design. We could actually put resources such as function units into hibernation, then we could run image processing programs, which give you courser images at their by safe power. We could do remote execution. Do not execute on the mobile unit, but execute the application on a remote server. And then, schedule the instructions, which I have already mentioned, memory management, turning of cache

and memory banks when no performances is needed and energy accounting. This we will see in detail when we actually look at the cloud and data center application.

(Refer Slide Time: 20:52)

Where does the power go?

		Storage	Communication I/O	Computation	Others
General Purpose Computing	Server	3 DRAM/DISK	2	1	Power Supply
	Work Station	2	WLAN/LCD	GPU/CPU	2
IoT for Digital Consumer	Receiver	2	1	1	2
	Transmitter (RF/Cellular)	2	3	2	1
	non-Communication	1	2	specialized HW	1
Very Low Power (Sensors)		2 may be 1 for non-RF hardware	3	2	1

1 - most important, 3 - least important
Y.N. Srikant

In other words, we really need to look at all aspects of a computer system, starting from hardware to software to application, in order to reduce the power consumption. How? Where does the power go? Which parts of a system consume more or less power? Let us see. And what type of a system, of course? At the bottom of this chart, we have very low power devices, such as sensors. Then, there are 4 columns, corresponding to storage, communication, and I/O, computation, and others.

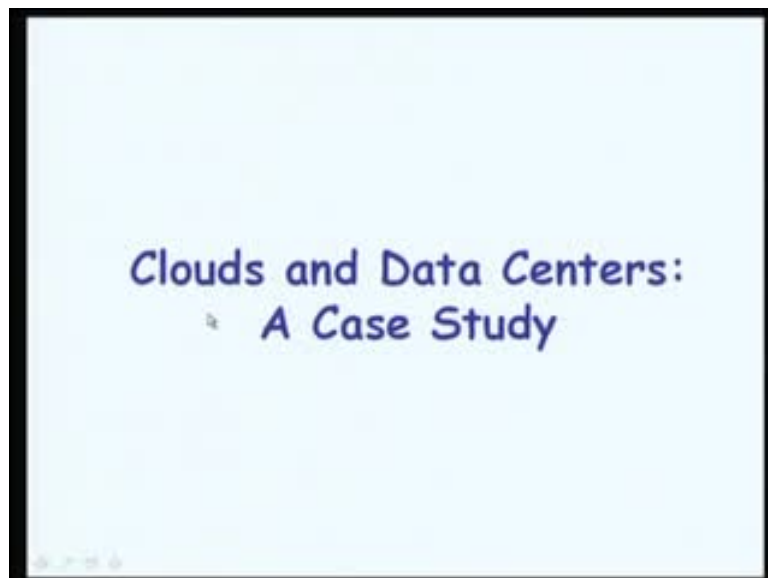
A sensor network system will have very little storage, so it is not very important to have more storage. **This could be 2.** For example, there is not too much storage on a low power sensor system, but if it is a non-RF, there could be a lot of storage. So that is why, it is 2 for sensor devices and for non-RF hardware, it is 1. That is most important to consume power. Communication is the highest power consumption operators on the sensor – the transmitter and receiver. Whereas, computation is least important, hence it does not take too much power to compute, whatever it is reading.

At the highest end, the server, the d ram and disk are the storage devices. It is important to conserve power in these devices. Communication, of course, is not so important in a server. However, computation is very important, so we need to have the CPU reduced power consumption. Of course, power supply is also needed to be designed appropriately

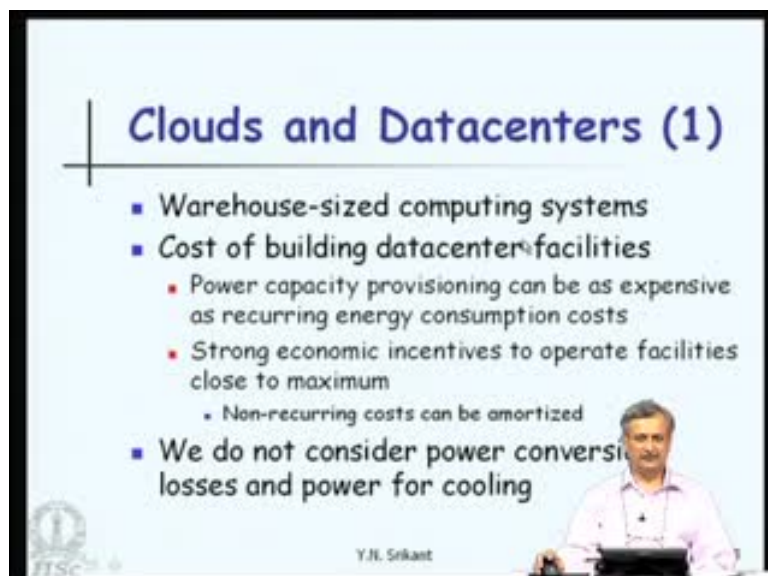
to reduce the power consumption. This is how the power consumption spectrum really looks like.

In a workstation, for example, rather, in a system on chip, etc., storage is very low, so we have to conserve power only in the communication. Whereas, if you look at no communication digital device, there may be a lot of storage. So, we need to reduce the power consumption in storage, application and the device, etc. It is important to see which one deserves attention for power consumption reduction.

(Refer Slide Time: 23:39)



(Refer Slide Time: 23:50)

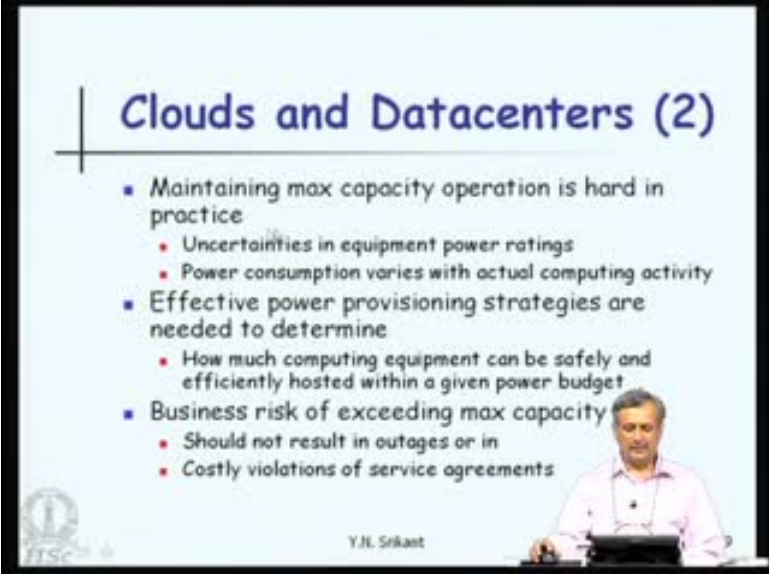


Let us take a simple case study of clouds and data centers. Clouds are nothing, but for us, the abstraction is a collection of computing systems – extraordinarily large warehouse-sized computing systems. Clouds and data centers are the same as far as we are concerned. There is a technical difference between the two but that is not important for us.

These are extraordinarily large – they would have possibly 4,000 to 8,000 servers or more. The cost of building data center facilities is very large. In fact, apart from the computing elements, we need to actually worry a lot about the power capacity provisioning as well. How much power capacity should we provide to a data center? Power capacity provisioning can be as expensive as recurring energy consumption costs. As we need to build generator, UPS station, and so on and so forth, this becomes very important. What happens is you could actually be providing a large capacity, but we may not be using all the capacity. So you have wasted a lot of money in building the power plant with the capacity which is more than what is needed. But, if you provide very little expansion possibility, then you may be actually getting saturated very soon and you will not be able to add more servers or more equipment, to your datacenter.

There are strong economic incentives to operate facilities close to the maximum because if you operate the power plant, etc., at the almost maximum level, non-recurring cost can be amortized. You can recover the non-recurring costs very quickly. In this study, we do not consider power conversion losses and power for cooling, that is, air conditioning, etc.

(Refer Slide Time: 26:00)



The slide is titled "Clouds and Datacenters (2)" and features a list of three main points, each with sub-points. A small inset image of a man in a light blue shirt is visible in the bottom right corner of the slide. The text is as follows:

- Maintaining max capacity operation is hard in practice
 - Uncertainties in equipment power ratings
 - Power consumption varies with actual computing activity
- Effective power provisioning strategies are needed to determine
 - How much computing equipment can be safely and efficiently hosted within a given power budget
- Business risk of exceeding max capacity
 - Should not result in outages or in
 - Costly violations of service agreements


Saying yes, we must use the power plant to its maximum capacity is easy, but maintaining maximum capacity operation is very hard in practice. Why? There are uncertainties in equipment power ratings. You may say something requires 60 watts, but actually it may be requiring only 15 to 20 watts when it is running on medium load. It may be requiring the full 60 watt only at very peak load, which is very infrequent. This is the problem. The manufacturer would have rated the equipment as 60 watts, but in reality, on an average, it may be running at 20 to 30 watts.

Power consumption varies with actual computing activity. This is another problem. So computer systems do not take the same amount of a power continuously. Their requirement varies with applications, which are running.

Effective power provisioning strategies are needed. Why? We need to determine how much computing equipment can be safely and efficiently hosted within a given power budget. So, putting very few computing equipment, based on the name plate, rather specification plate figures, would be risky because you are going to put very few of them and power plant will not be used to its maximum capacity. Whereas, if you put too many, then there may be outages and this will create a service-level / service agreement violation and this may be very expensive.

We cannot run the business of exceeding the maximum capacity. We should be very careful and judicious in planning how many computing elements we can put in the datacenter?

(Refer Slide Time: 28:22)



Determining right deployment and power management strategies

- Requires understanding simultaneous power usage characteristics of 1000's of machines over time
- Important factors
 - Rated max power of computing equipment is overly conservative and not useful
 - Actual consumed power of servers varies significantly with activity
 - Different applications exercise large-scale systems differently
- Hence, only monitoring of large-scale workloads can yield insights into the aggregate load at the datacenter level

Y.N. Srikanth

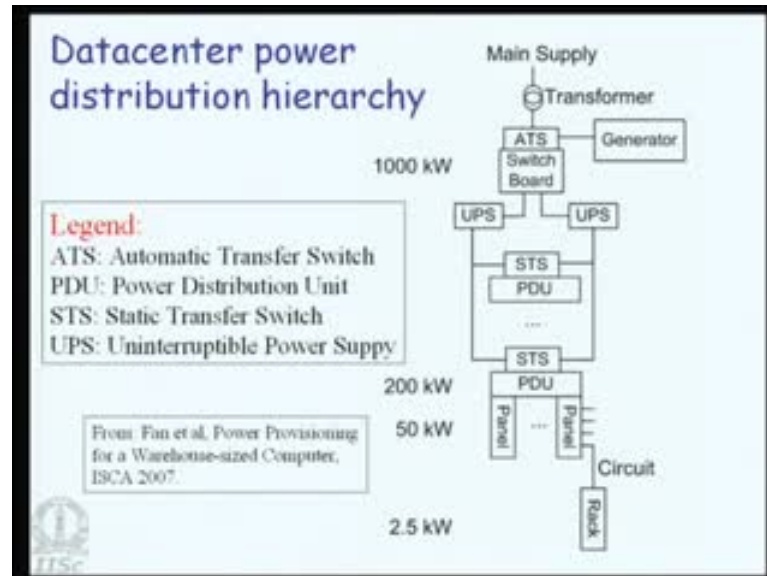
How do we determine the right deployment and what power management strategies are we going to use? This requires understanding simultaneous power usage characteristics of thousands of machines over time. A datacenter would have thousands of machines, so it is very difficult and almost impossible to learn the characteristics of every machine. Well, some machine may fail and it may be replaced as well, so it becomes almost impossible to know the exact characteristics – power usage characteristics – of all the machines.

What are the important factors that we need to keep in mind? The rated maximum power of computing equipment is overly conservative and not useful. I have already mentioned this. Whatever is mentioned on the plate outside the equipment is an overwriting, so we cannot use this for counting the number of equipment.

Actual consumed power of servers varies significantly with activity. Applications determine this amount. Different applications exercise large-scale systems differently. This is another issue. Hence, we need to monitor large scale workloads and only then we can get insights into aggregate load at the datacenter-level. Unless large scale workloads

are monitored, you cannot really get ideas on how to control the power consumption in a system. Running small applications on a single CPU, etc., will not give us this insight.

(Refer Slide Time: 30:06)



This is the datacenter power distribution hierarchy. We have a main supply, a transformer, a switch, which either switches to the main supply or to the generator. Then we have a number of UPS, static transfer switches to switch the power distribution unit to either this UPS or this UPS. You observe that at the highest level, power capacity is 1000 kilo watts. As we go down to the power distribution unit, each one of them can handle 200 kilo watts. The PDU supplies panels, each of which can handle 50 kilo watt. Panels, in turn, supply to racks, each of which can handle 2.5 kilo watts, and the racks will contain the motherboards, each of which may require 300 to 500 watts or more.

(Refer Slide Time: 31:10)

Inefficient use of power budget

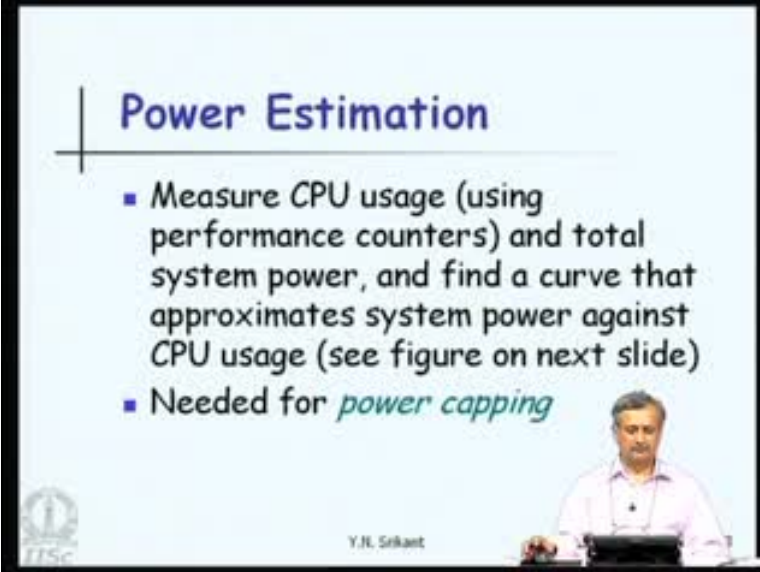
- Staged deployment
 - Facility is sized to accommodate business demand growth
- Fragmentation
 - Addition of one more unit might exceed that level's limit and such unused capacities add up at the datacenter level
- Conservative equipment ratings (60% more)
- Variable load
- Statistical effects
 - It is unlikely that large groups of systems will be at their peak activity (therefore power) levels as the size of the group increases

Y.N. Srikanth 22

Inefficient use of power budget is a problem – we may have staged deployment. In other words, facility is sized to accommodate business demand growth. We do not use the entire power plant, we add only few units and as our requirement goes up, we really put more and more units into operation.

Fragmentation: Addition of one more unit might exceed the levels' limit. For example, at the rack-level, you may not be able to add one more board because the rack capacity of 2.5 kilo watt may get exceeded. That is by looking at the specifications of that motherboard. But in reality, each motherboard may be taking quite a bit less compared to what is this specification. So, there is unused capacity even at the rack-level. For example, if you have put 5 boards, there may be possibility of putting 6 or 7 boards. That means, a little bit of power capacity has not been used at the rack-level. Similarly, at each rack-level, this can accumulate, at the PDU-level it becomes bigger, and finally, there may be a lot of capacity in the system, which is unutilized but fragmented up to the rack-level. There is the conservative equipment rating, which I have mentioned already. It is actually 60 percent more than what is really used – variable load and statistical effects. For example, statistically it is known that it is unlikely that large groups of systems will be at their peak activity levels as the size of the group increases. In other words, if there are thousands of processors, not all thousand processors will run at the peak capacity level.

(Refer Slide Time: 33:12)



Power Estimation

- Measure CPU usage (using performance counters) and total system power, and find a curve that approximates system power against CPU usage (see figure on next slide)
- Needed for *power capping*

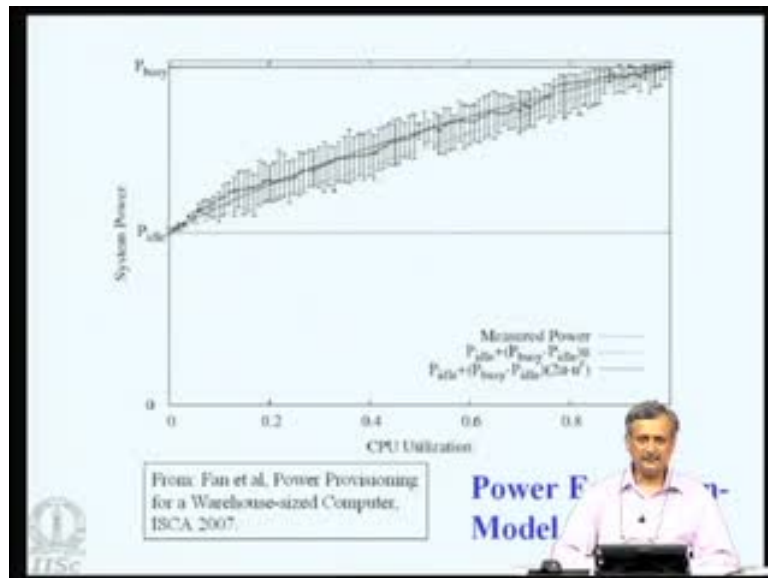
Y.N. Srikant

The slide features a light blue background with a black border. The title 'Power Estimation' is in a dark blue font. Two bullet points are listed below the title. The second bullet point includes the phrase 'power capping' in italics. In the bottom right corner, there is a small inset image of a man with a beard, wearing a light-colored shirt, sitting at a desk with a laptop. The name 'Y.N. Srikant' is printed below the inset image. A small logo is visible in the bottom left corner of the slide.

How do we estimate power? We measure the CPU usage, using performance counters, and the total system power, that is, using some meters, and so on, and find a curve that approximates system power against CPU usage. What we want to tie is the CPU usage to the total system power? This is necessary for power capping, which I will explain a little later.

In other words, we want to actually measure the CPU usage, which is very easy to do by software using performance counters. Once we have a graph and an equation which tells you what is the system power for each CPU usage, it can be used to our advantage. We do not have to measure system power all the time.

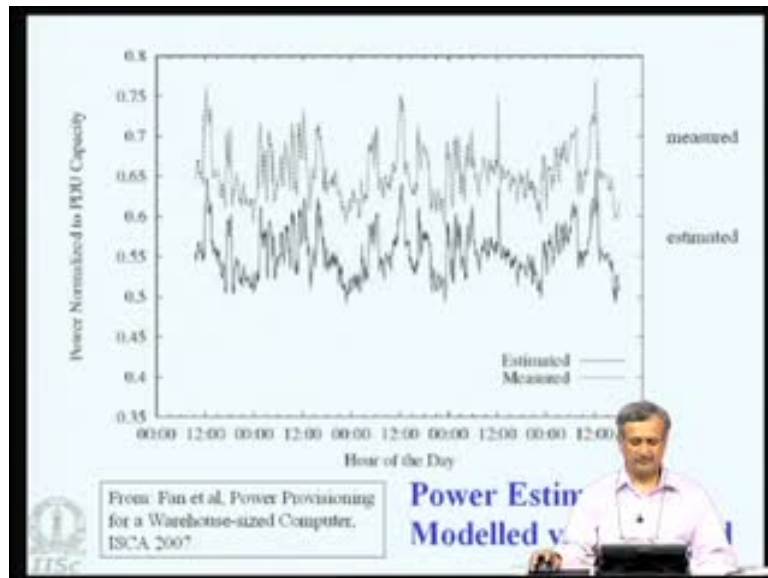
(Refer Slide Time: 34:19)



Here is the graph. This is the idle power level and this is the busy power level for the system. In between, the power utilization of the system actually varies. These bars, vertical bars, as we see, are the actual measurements that we have taken. And in between, we have a line here. At the lower level, we have a dotted line, and in the middle of these bars, we have a solid line. The solid line and the dotted line correspond to two models, which have been fitted to this particular usage pattern.

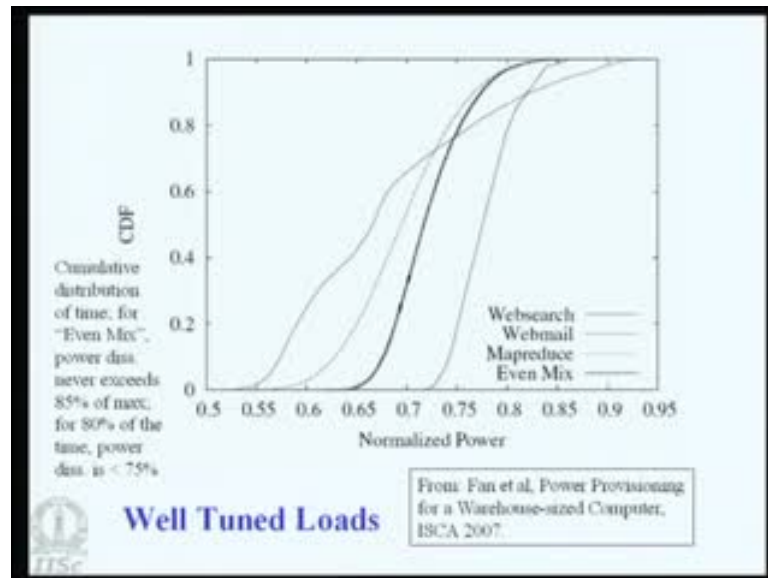
This has CPU utilization on this end and system power consumption on the other side. One of the models – the solid line model – seems very accurate. It actually reflects how the power is consumed within the system. So let us see whether this tallies with the actual system and measurements.

(Refer Slide Time: 35:29)



Here are two graphs. One of them is the measured power consumption in the system and the other is the estimated power consumption in the system. Using the fitted model, that is, the equation that we saw in the previous slide, you can see that the measured power consumption is higher than the estimated one, but only by a fixed offset. That offset is same at all points in time. So, that can be easily adjusted. The moral of the story is, by using that equation, which we have presented in the last slide, we are able to actually come up with a very accurate estimate of power consumed by the whole system, as a function of CPU usage.

(Refer Slide Time: 36:20)



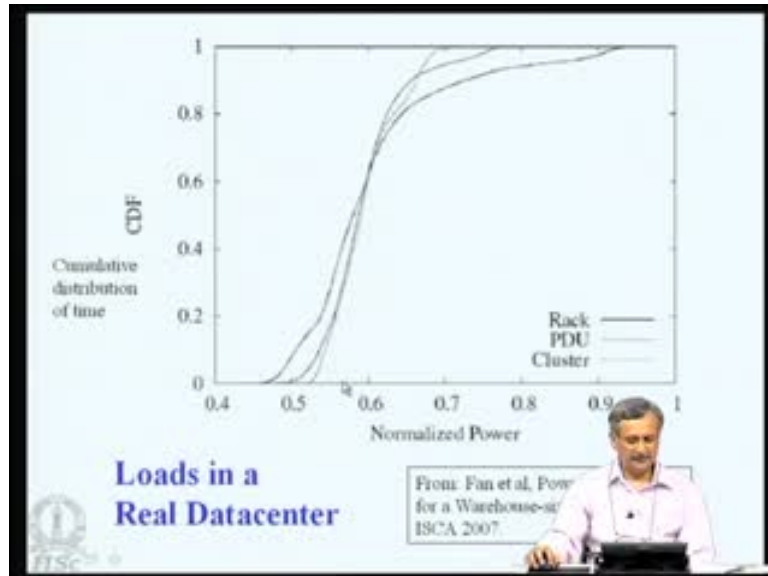
Let us also see how much power is used on an average in a system and running applications. On this x-axis, we have normalized power consumption, and on the y-axis, we have cumulative distribution of time. What is the implication? How do we read these graphs? At the lowest level, when we are yet to begin, that is, we are just idling; the normalized power consumption is still about 0.5 or so. In other words, this is the idle power consumption of the system, when the system is doing nothing. As we go on executing the program, we will reach one, that is, the end. During this transition from 0 to 1, that is, we actually see that the power consumption of various applications – web search, web mail, map reduce, etc., vary. Let us look at a mix of these applications. We start with 0.65 as idle power consumption, and 60 percent of the time, we would be using only about 0.72 or 72 percent of peak power consumption of the system.

Even when we reach 1, that is, let us assume that there are portions of the program which actually execute at the peak power, this is the peak power. The peak power is only 85 percent of the maximum power that we assumed we have. So, the rating is 1 but we have used 85 percent. For most of the time, say 95 percent of the time, the applications run at less than 80 percent peak capacity. So, 20 percent is the over provision that we have on the computer system.

In other words, we could put another CPU, so that one more program can run on that particular CPU and the power consumption. If we add one to four or something like 20

percent saving in each one of them, it will give us one more CPU, which can be added. This is the way we use this graph to determine whether we should add something or we should not add something.

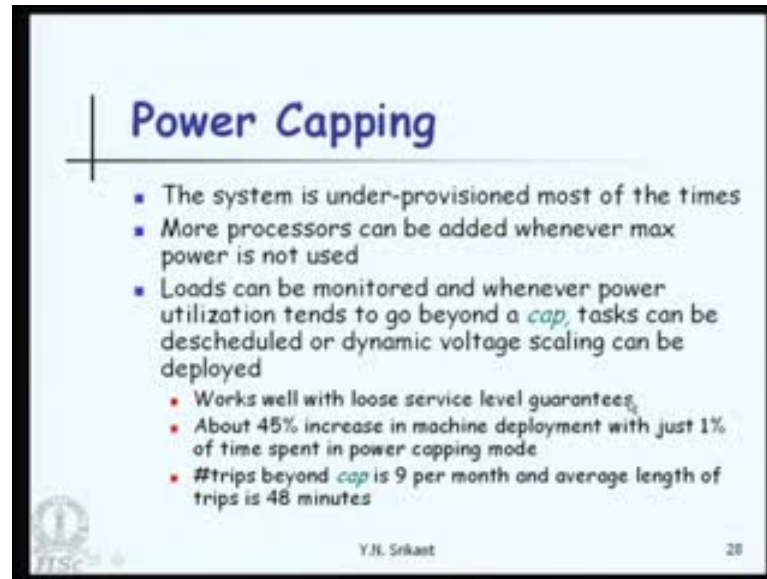
(Refer Slide Time: 39:45)



When should we not add something? Consider this particular graph. This really corresponds to web search. In web search, there are parts of the program when the power consumption system is almost 95 percent. In other words, you cannot take the risk of adding another CPU, if only web search is running on all the machines. If there is a mix of these programs then may be, for four CPUs, you can add an extra CPU, etc.

This is a similar graph. So, normalized power with rack PDU and cluster-level for various applications the mix. If you look at it, this is the cluster-level. See that cluster-level is over-provisioned – 30 percent more. It really consumes only 70-72 percent. Most of you know the highest level. There are no programs, which really require it to consume 100 percent of the power. That is possible.

(Refer Slide Time: 40:23)



Power Capping

- The system is under-provisioned most of the times
- More processors can be added whenever max power is not used
- Loads can be monitored and whenever power utilization tends to go beyond a *cap*, tasks can be descheduled or dynamic voltage scaling can be deployed
 - Works well with loose service level guarantees
 - About 45% increase in machine deployment with just 1% of time spent in power capping mode
 - #trips beyond *cap* is 9 per month and average length of trips is 48 minutes

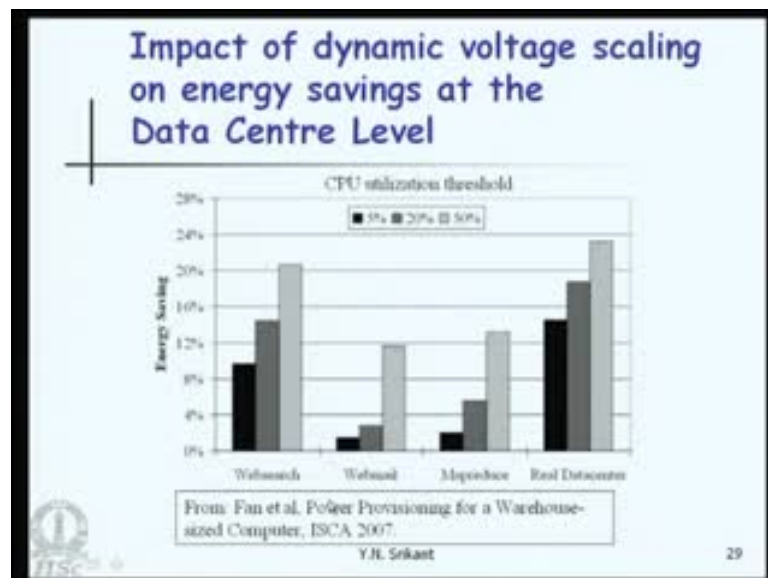
Y.N. Srikanth 28

Now, having said all this, we understand that the CPU runs programs, the datacenter runs programs of all kinds. Not all CPUs are going to run at the peak power level, and therefore, there is a possibility of adding one more CPU and run more programs. Power is saved because each CPU requires less power than what its rating says, may be 70 percent is all that it is using, 30 percent is spare. So, we could add one extra CPU for each of the 3 CPUs in the system. But suppose, we add more CPUs and suddenly the processors start executing programs which require high power consumption. That is possible. In such a case, we could be exceeding the power consumption of the system beyond the rating. So the system is under-provisioned most of the time. More process can be added whenever maximum power is not used, but then loads can be monitor and whenever power utilization tends to go beyond a cap, that is, whatever threshold we set, task can be descheduled or dynamic voltage scaling can be deployed. In other words, you reduce the voltage of the entire data center – all CPUs. That will reduce the power consumption of all CPUs and you can work at a lower frequency and lower performance level for certain duration. Once some of the task complete and exit, perhaps the situation will change, and you can increase the voltage and the frequency of the system again.

This works well with loose service-level guarantee. There are no hard service-level guarantees, that is, I am going to finish this particular job in such an amount of time. Such guarantees are not really provided. So, give or take an hour, your program runs to completion, the customer is happy. About 45 percent increase in machine deployment

with 1 percent of time spent in power capping mode. In other words, the system actually starts using peak or little more than peak power only for 1 percent of the time. In that 1 percent of time, you reduce the voltage and immediately the system finishes some jobs and goes back to low power or underutilize mode again. This is really what happens in practice. Because of that we have really put 45 percent more CPUs, and only 1 percent of the time, the machine has gone to peak power condition. Only 1 percent of the time is spent in power capping mode, that is, when the peak power condition is met and number of trips beyond the caps is only 9 per month -- only 9 times. Average length of each trip is only 48 minutes. If you have a loose service-level agreement, this is hardly the price to pay because you are using the power provisioning of the plant to maximum capacity. The customer is also going to be charged less. You are going to have more customers, so the cost per customer will be lesser.

(Refer Slide Time: 44:12)



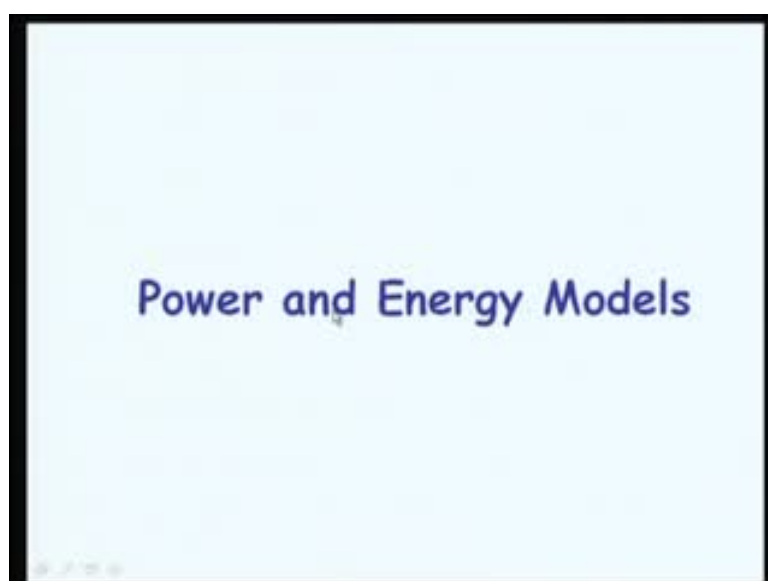
What is the impact of dynamic voltage scaling on energy savings at the data center-level? I have already mentioned that we are going to use DVS at the data center-level. What we really have are 3 sets, you know; 4 sets of graphs for different application, 3 of them for different application and on a real data center with a mix of applications and on a real data center with a mix of application this is for the fourth one. We have three bars in each of these groups -- one says 5 percent, next one says 20 percent, and 50 percent CPU utilization threshold. What does this really mean?

On this side we have energy saving. So, 5 percent means I cannot tolerate -- very slow CPU performance. At some point in time, let us say, I have reduced the voltage dynamic, voltage scaling has happened. I have reduced the voltage of the data center, the CPU utilization will come down by 5 percent. Then, I cannot tolerate any more. I am going to increase the voltage back to its normal.

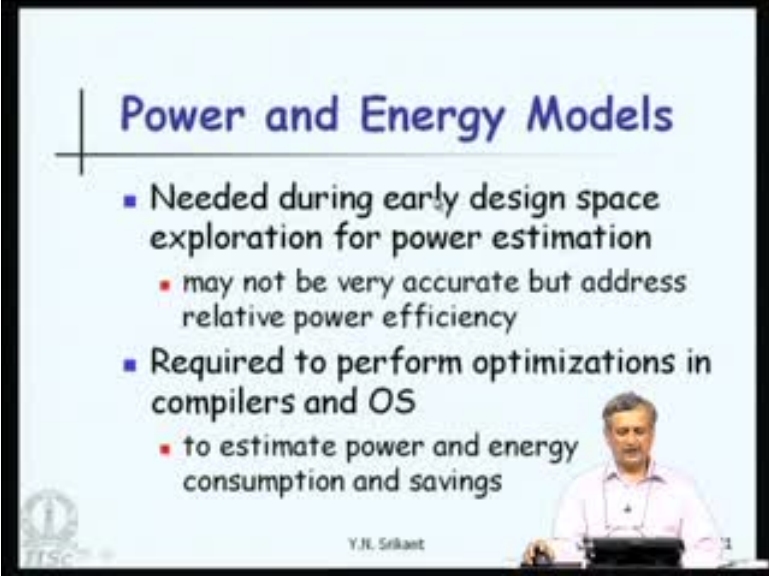
Obviously, if this threshold is only 5 percent, then I will be running the data center, at most of the time, at the highest voltage level and for only a very small number of times I am really going to reduce the voltage. So, the saving is quite less. It could be about 8-10 percent. Whereas if I can tolerate 20 percent slowdown in my CPU speed, in other words, my application takes 20 percent more time, then I get more -- may be 15 percent or so. And if I can tolerate 50 percent slowdown in my application, I can actually get much more saving -- 20 percent or more.

But obviously, we cannot sacrifice performance to such a great extent, so we may be somewhere between 5 and 20 percent slow as far as DVS is concerned. Normally, it is not more than 5-6 percent. Later, we are also seeing compiler techniques to do dynamic voltage scaling. Then, we can do fine grind control and get much more power saving than what is mentioned here. This is actually at the data center-level. The operating system would reduce the voltage of the entire data center and not on CPU basis.

(Refer Slide Time: 46:58)



(Refer Slide Time: 47:18)



Power and Energy Models

- Needed during early design space exploration for power estimation
 - may not be very accurate but address relative power efficiency
- Required to perform optimizations in compilers and OS
 - to estimate power and energy consumption and savings

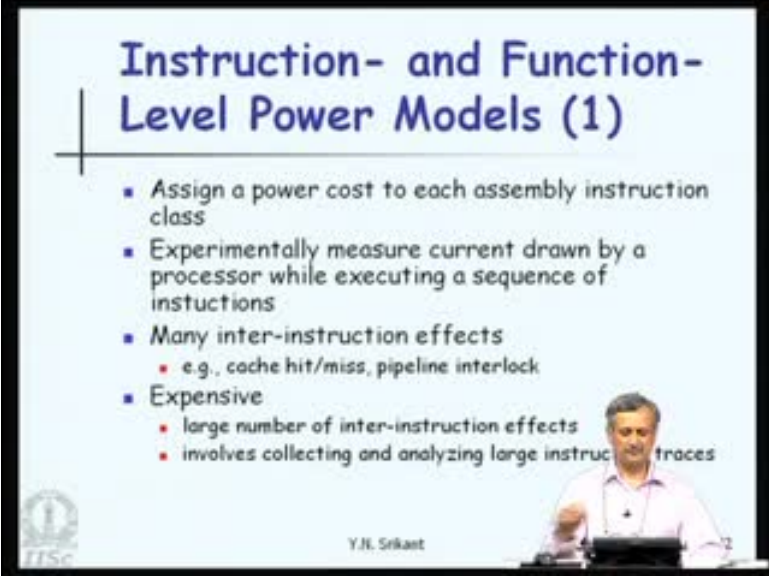
Y.N. Srikant

Now, we consider power and energy models. In the first part, we considered the introduction and then we saw cloud and data center application in power saving. Why do we need power and energy models? We need power and energy models during early designs space exploration for power estimation. We have started designing a system and we need to know how much power our system is going to consume when it is implemented. Why? Why do we need such an estimate?

Well, we want to design alternative designs, which can save energy. Unless we know how much power each one of these designs will take, we really cannot decide which design to implement. At this level, the models may not be very accurate because we do not have all the data about the implementation. They address relative power efficiency. Is design A better than design B? That is going to be the question that we answer. This can be answered reasonably well with the power and energy models that we are going to study.

They are also required to perform optimizations in compiler and operating system. Compilers and OS require models to estimate power and energy consumption, and thereby they use strategy a, b, c, etc., in order to save energy. OS and compilers also require such models.

(Refer Slide Time: 48:50)



Instruction- and Function-Level Power Models (1)

- Assign a power cost to each assembly instruction class
- Experimentally measure current drawn by a processor while executing a sequence of instructions
- Many inter-instruction effects
 - e.g., cache hit/miss, pipeline interlock
- Expensive
 - large number of inter-instruction effects
 - involves collecting and analyzing large instruction traces

Y.N. Srikant

Let us look at the first model – instruction- and function-level power models. At the instruction-level, we are going to assign a power cost to each assembly instruction class. In other words, the entire set of ALU instructions – add, subtract, etc., each of these will be assigned the same cost. Each class of instructions, which uses ALU or floating point unit or memory or something else, will be assigned a cost.

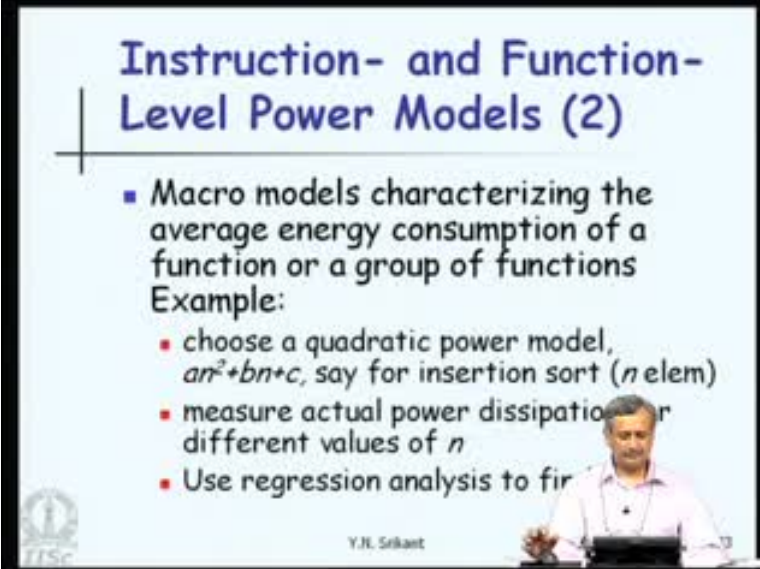
Each instruction in that class will be assumed to consume the same amount of power. Then, how do we assign this cost? We experimentally measure current drawn by a processor while executing a known sequence of instructions. We try to put known sequence – the same kind of instructions – execute, make the program run on a processor, then look at the current which is drawn by the processor and compute the power by multiplying it by the voltage.

Such a simple scheme has problems. There are many inter-instruction effects. For example, cache hit miss. If the data or instruction is in a cache, the power consumption is different as compared to when it is in the main memory and not in the cache. Then, there could be pipeline interlocks. If there is a delay and the load is not yet over, there is a pipeline stall. This could actually cause different power dissipation in the system.

To take care of such inter-instruction effects, we really have to run a very large number of programs in order to assign cause to the instruction. The problem is – there are a large number of such inter-instruction effects, so the multiplicative effect would be large. We

will have to collect and analyze extremely large number of instruction traces, in order to assign reasonable cost to instructions.

(Refer Slide Time: 51:32)



The slide is titled "Instruction- and Function-Level Power Models (2)". It contains a bulleted list of macro models characterizing average energy consumption. An example is provided for insertion sort, which is modeled as a quadratic power model $an^2 + bn + c$. The example steps are: choose a quadratic power model, measure actual power dissipation for different values of n , and use regression analysis to fit the model.

- Macro models characterizing the average energy consumption of a function or a group of functions

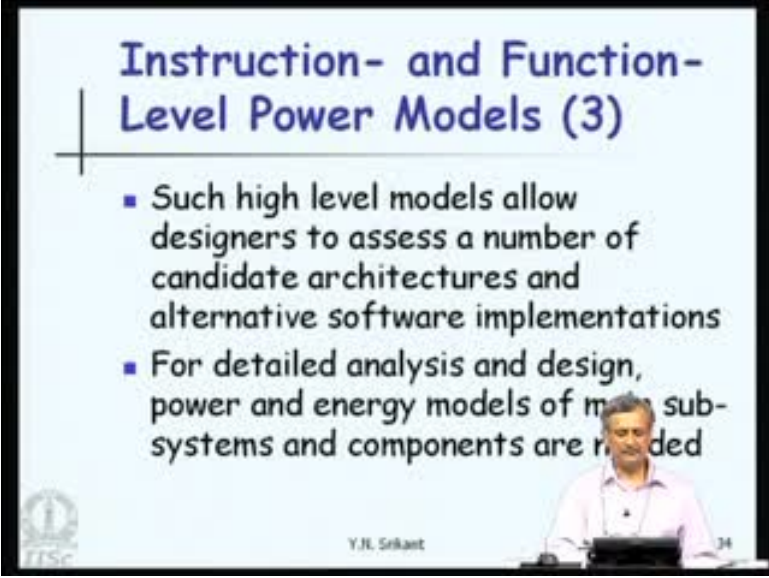
Example:

- choose a quadratic power model, $an^2 + bn + c$, say for insertion sort (n elem)
- measure actual power dissipation for different values of n
- Use regression analysis to fit

Y.N. Srikant

This model is not very easy to, let us say, make and calibrate. Suppose we want to do it on a program-level or function-level, we can have macro models characterizing the average energy consumption of a function or a group of functions. For example, suppose we know that we are running insertion sort, we know that the time required for insertion sort is a_n square, where n is the number of elements. So, we could choose a quadratic power model for the power consumption, say a_n square plus b_n plus c for the insertion sort program.

(Refer Slide Time: 52:35)



Instruction- and Function-Level Power Models (3)

- Such high level models allow designers to assess a number of candidate architectures and alternative software implementations
- For detailed analysis and design, power and energy models of main subsystems and components are needed

Y.N. Srikant 34

Measure the actual power dissipation for different values of n and then use regression analysis to find a , b , c . Given n , we can say what the power consumption for that particular n . This is a way of characterizing the energy consumption of a program. Such high-level models allow designers to assess a number of candidate architecture and alternative software implementations. In an embedded system, the same program or same set of programs will be run again and again. In the washing machine, the same program runs again and again. Or in a media player also, the same thing happens. The program is known, and if we build the approximate models like what we saw just now for this particular program on various architectures, then we can access and say this architecture is better for this program or this architecture is very bad for this particular program, etc. Then, we can also look at alternative software implementation insertion sort, may be, written in different ways and you can similarly compute, rather have a model for each one of this implementations, test it on various architectures, and choose the correct candidate architecture for your application.

However, for very detailed analysis and design, that is, design of the hardware, power and energy models of main subsystems and components are also needed. In other words, we will need micro-architectural-level, ALU-level, memory-level, cache-level inter connection-level models, in order to study the design and make an estimate of how much power is consumed by that particular design. We will stop at this point and continue with such micro-architectural models in the next lecture. Thank you.