Very common question: *Is something wrong with the system (machine)?* It's the job of the Operations group to rule out a system issue, before getting engineers to start poking at their code (barring an obvious code issue ;)).

CPU utilization:

`uptime examples`:

```
 23:04:33 up 59 days, 10:35, 10 users,  load average: 0.00, 0.02, 0.05
 21:42:30 up 460 days,  7:47, 10 users,  load average: 0.01, 0.03, 0.05
 21:32:15 up 1891 days,  5:16, 11 users,  load average: 0.03, 0.04, 0.00
```

The last three numbers are the 1min, 5min, and 15min "load average". Generally speaking, the load average is the average number of processes in the system "run" (or "ready") queue, so when there was only 1 CPU, a load average of "1" meant that the system was fully occupied. Load average *used to* be a very good measure of actual system load, back when CPUs were all single-core. Now, with multiple CPUs, you could have a load average of "4.00" and this means that on a quad-core system, your system is fully occupied, whereas on a single-CPU system, it means the system is (technically) overloaded 4x. If you are able to take into consideration things like CPU count, then "load average" is still a pretty good measure of how (over)loaded a system is.

Some systems (Linux in particular) will also show any processes that are blocked due to I/O, which can artificially increase the "load average".

Other utilities can describe the system a little better. A really good one is "vmstat", which looks like this:

`vmstat`:

```
procs -----------memory---------- ---swap-- -----io---- --system-- ----cpu----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in    cs us sy id wa
 1  0 332532 136728 140328 2836252    0    0     0    24 1078   211  0  0 100  0
 1  0 332532 136728 140328 2836252    0    0     0     0 1109   270  0  0 100  0
 0  0 332532 136456 140328 2836252    0    0     0    72 1146   340  0  1 99  0
 0  0 332532 135084 140344 2836264    0    0     0   648 1301   667  2  2 89  7
 0  0 332532 134216 140344 2836268    0    0     0   220 1199   414  3  2 94  1
 0  0 332532 134216 140344 2836268    0    0     0    20 1061   155  0  0 100  0
```

This gives a much better view of what, exactly, is going on in the system -- and gives you a bit of a "historical" view (assuming you leave it running for a bit). The "r" and "b" columns actually "describe" the load average (processes that are [r]unning or [b]locked), (we'll cover memory later), the "io" section shows Blocks In and Blocks Out, the "system" section shows INterrupts and Context Switches, and the last section, "cpu", shows up overall processor states (USer, SYstem, IDle, WAit) in percent.

`'top'` shows similar info at the top of its output, like:

```
top - 20:56:01 up 208 days, 15:11, 28 users,  load average: 0.23, 0.16, 0.10
Tasks: 358 total,   1 running, 355 sleeping,   1 stopped,   1 zombie
Cpu0  :  0.5% us,  1.0% sy,  0.0% ni, 96.1% id,  2.0% wa,  0.0% hi,  0.5% si
Cpu1  :  1.0% us,  0.5% sy,  0.0% ni, 98.5% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu2  :  4.9% us,  2.0% sy,  0.0% ni, 93.1% id,  0.0% wa,  0.0% hi,  0.0% si
Cpu3  :  0.5% us,  1.0% sy,  0.0% ni, 91.6% id,  6.9% wa,  0.0% hi,  0.0% si
Mem:   4145484k total,  4018024k used,   127460k free,   139776k buffers
Swap:  3148700k total,   332952k used,  2815748k free,  2853500k cached
```

Following that, you get a list of processes, usually sorted by CPU usage:

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 9159 apache    20   0 26808  12m 1576 R  8.0  0.3   0:00.91 /usr/local/apache2/bin/httpd -k start
16103 dovenull  20   0  3928 2272 1724 S  1.5  0.1   0:00.03 dovecot/imap-login [1 connections (1 TLS)]
14021 root      20   0  2200 1164  760 R  1.0  0.0   0:39.33 top
 2186 taner     20   0  8668 4456  672 S  0.5  0.1 201:03.73 SCREEN
16094 root      20   0  2608 1340  984 S  0.5  0.0   0:00.02 dovecot/auth worker: idling
18561 root      20   0  2696  996  720 S  0.5  0.0   0:43.06 /usr/sbin/dovecot
    1 root      20   0  1604  576  524 S  0.0  0.0   0:49.28 init [5]
    2 root      15  -5     0    0    0 S  0.0  0.0   0:00.03 [kthreadd]
```

Now that we've looked at what is consuming CPU cycles, it's important to see how much memory is in use.

We can use both vmstat and top, as well as the 'ps' command to do that. The "top" program is really powerful, as you can add other data you want to track - for example, page faults can help you track down a process that might be swapping often.

After turning on page fault count, and then sorting by page faults, we see:

```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  nFLT COMMAND
 2523 mysql     20   0  137m  13m 2260 S  0.0  0.3 800:34.65 7766 /usr/libexec/mysqld --defaults-file=/etc/my.cn
 2824 mailman   20   0 19388 8200 1952 S  0.0  0.2  18:24.27 4976 /usr/bin/python /usr/lib/mailman/bin/qrunner -
 2704 squid     20   0 10132 5452 1540 S  0.0  0.1   0:44.08 3701 (squid) -D
```

...as I had expected, my mysql process has the most swap activity, which is not unexpected, given that it's one of the larger running processes (and this system is slightly memory constrained).

'ps' under Linux has two sets of command line switches you can use, the BSD syntax ("ps aux"), or "standard" syntax ("ps -eF") -- both methods will get you similar results, just be aware that there are slight differences ;) 'ps' can also be useful to also check general process info -- like how much CPU the process has consumed, memory usage, and things like the actual command line it was called with, as well as the Process ID (PID) and Parent PID (PPID). If PPID is '1', this means the parent is gone, and now process "1" ("init") is the parent... while it can be normal for there to not be a running parent process, it can be really useful to use the PPID info to find related processes.

I highly recommend playing around with these commands -- checking the 'man' pages for each, etc.

In addition to checking CPU and Memory, you'll want to take a look at I/O. This is primarily Disk activity, but also Network.

For disks, the best (common) tool is likely 'iostat'. My system is not very loaded, so the iostat output is pretty boring:

```
avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.70    0.00    0.85    0.50   97.95

Device:            tps   Blk_read/s   Blk_wrtn/s   Blk_read   Blk_wrtn
sda               5.39         0.00        75.05          0        376
sda1              0.00         0.00         0.00          0          0
sda2              0.60         0.00         4.79          0         24
sda3              3.59         0.00        39.92          0        200
sda4              0.00         0.00         0.00          0          0
sda5              0.40         0.00        12.77          0         64
sda6              0.00         0.00         0.00          0          0
sda7              0.00         0.00         0.00          0          0
sda8              0.80         0.00        17.56          0         88
```

iostat can be invoked to scroll like vmstat does (updating every N seconds), and you can invoke it with the '-x' switch to get even more detail about the disk activity (avg size of requests, request service times, etc).

```
avg-cpu:  %user   %nice    %sys %iowait   %idle
           0.77    0.00    0.60    0.70   97.93

Device:    rrqm/s wrqm/s   r/s   w/s  rsec/s  wsec/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await  svctm  %util
sda          0.00   3.70  0.60  7.69    7.19   91.11     3.60    45.55    11.86     1.08  130.65   7.90   6.55
sda1         0.00   0.00  0.00  0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sda2         0.00   0.00  0.00  0.40    0.00    3.20     0.00     1.60     8.00     0.01   26.00  13.00   0.52
sda3         0.00   1.10  0.00  3.00    0.00   32.77     0.00    16.38    10.93     0.01    2.80   2.53   0.76
sda4         0.00   0.00  0.00  0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sda5         0.00   0.00  0.00  0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sda6         0.00   0.50  0.00  0.20    0.00    5.59     0.00     2.80    28.00     0.00   16.00  16.00   0.32
sda7         0.00   0.00  0.00  0.00    0.00    0.00     0.00     0.00     0.00     0.00    0.00   0.00   0.00
sda8         0.00   2.10  0.60  4.10    7.19   49.55     3.60    24.78    12.09     1.06  226.04  12.26   5.75
```

(Lecture time permitting – preview for next week)

For the network side of things, you want to look at things like:

- Connections
- Traffic (bps, pps)
- Packet loss / network routing

You'll want to use 'netstat' to check connections: are there more than expected?
Are they lingering?
Is a certain host trying to connect more than expected (malicious? unintentional?)?
One of the great options for netstat is "`-s`", which gives you a summary of a LOT of useful IP, TCP, and UDP data.
Here's an excerpt from the "TCP" section:

```
Tcp:

    6904 active connections openings
    55223 passive connection openings
    156 failed connection attempts
    4924 connection resets received
    6 connections established
    2057050 segments received
    1530644 segments send out
    1335 segments retransmited
    254 bad segments received.
    93295 resets sent
```

Some of the more useful numbers on the debugging side would be "failed connection attempts" and the "resets" info (well, and "bad segments", too :)). The "failed attempts" is a measure of how many times the local machine tried to connect to something else, and failed. Here's an example of how that increments:

```
root@ka:~# netstat -s | grep "failed connection attempts"
    156 failed connection attempts
root@ka:~# telnet localhost 9999
Trying ::1...
Trying 127.0.0.1...
telnet: Unable to connect to remote host: Connection refused
root@ka:~# netstat -s | grep "failed connection attempts"
    158 failed connection attempts
```

Even though I only attempted one telnet connection, the count incremented by two, because it first tried to connect to the IPv6 address (`::1`), and then to the IPv4 address (`127.0.0.1`).

Measuring actual traffic is a little harder with "standard" tools, but hopefully you can either install useful ones, or write some scripts to do the work for you. Strictly measuring interface speeds can be done with a script that looks at `/proc/net/dev` and calculates the difference in the numbers.

Programs like `iftop` and `iptraf` are a little more useful as they can show you exactly what connections there are, and which ones are doing the most traffic, etc. (These are both a little hard to show examples of, as they are interactive programs).

The program `tcpdump` is a bit more "raw" than `iftop/iptraf`, but that can be useful at times, to see actual packet-level information.