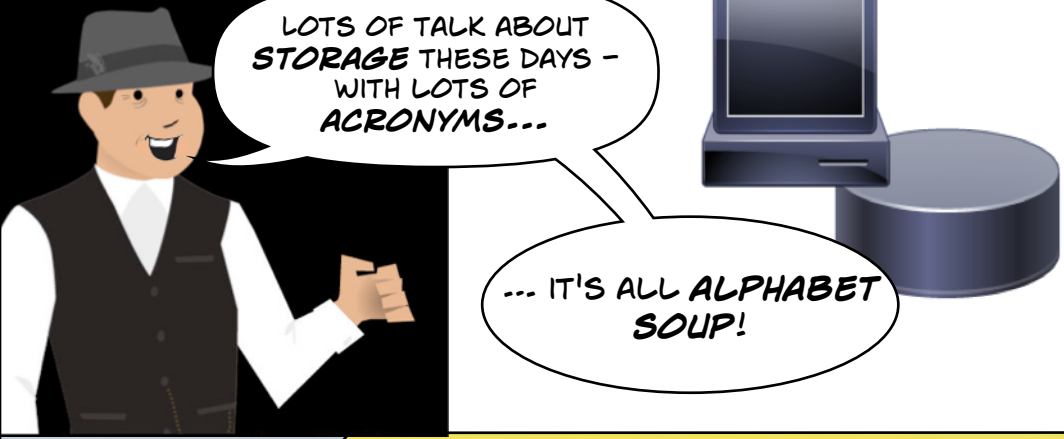
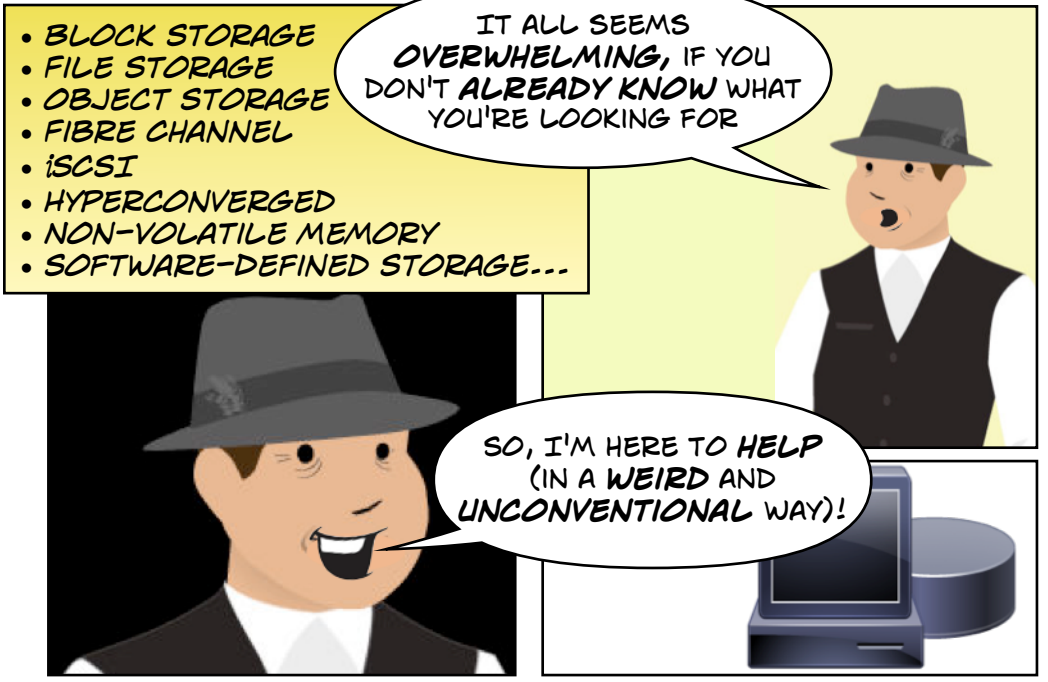
HAVE YOU EVER WONDERED...

ISSUE #5

NAPKIN DIALOGUES

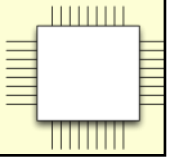


HOW STORAGE **REALLY** WORKS?


• BLOCK STORAGE
• FILE STORAGE
• OBJECT STORAGE
• FIBRE CHANNEL
• iSCSI
• HYPERCONVERGED
• NON-VOLATILE MEMORY
• SOFTWARE-DEFINED STORAGE...
IT ALL SEEMS OVERWHELMING, IF YOU DON'T ALREADY KNOW WHAT YOU'RE LOOKING FOR
SO, I'M HERE TO HELP (IN A WEIRD AND UNCONVENTIONAL WAY)!

I COULD SEND YOU TO A BUNCH OF **WHITE PAPERS**, **TEXTBOOKS**, OR **BLOGS**, BUT WHERE WOULD THE **FUN** BE IN THAT?

BESIDES, WHEN YOU'RE TALKING ABOUT...

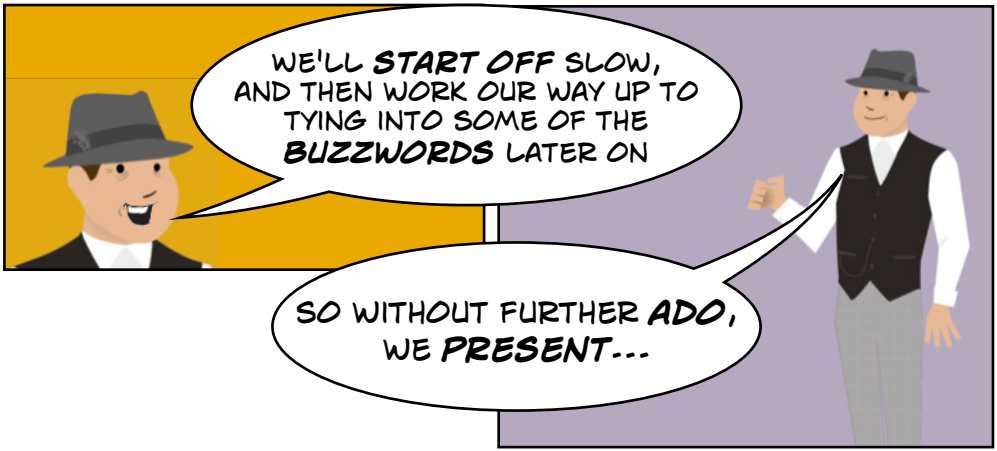


CPU...

MEMORY...

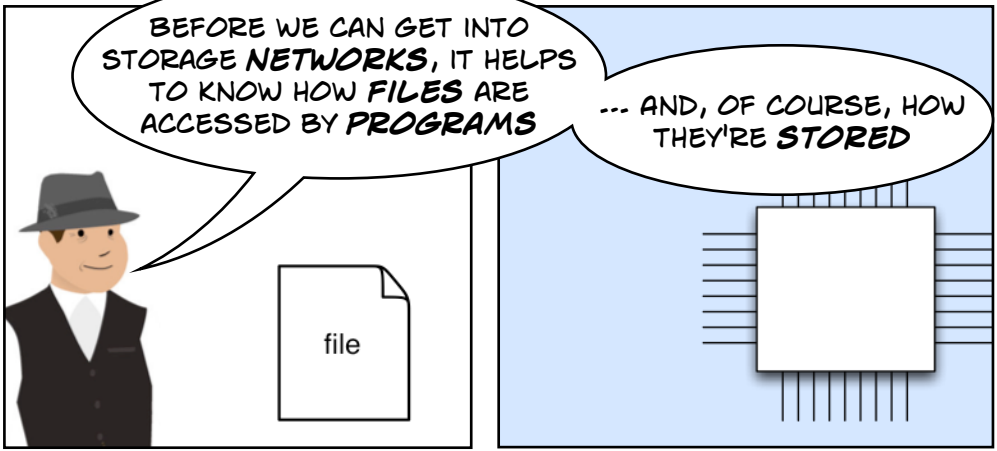
AND STORAGE TYPES...

IT'S EASIER TO **SHOW** EVERYTHING.



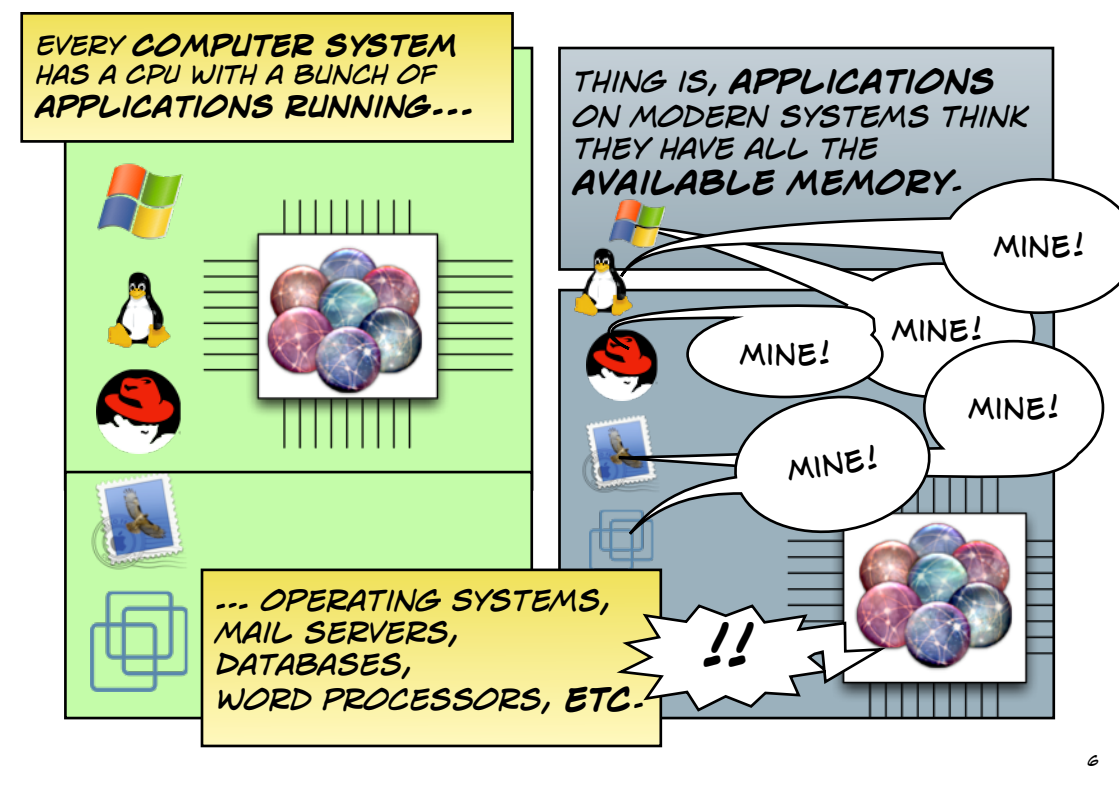
THE LIFE OF A STORAGE PACKET

NAPKIN DIALOGUES EDITION

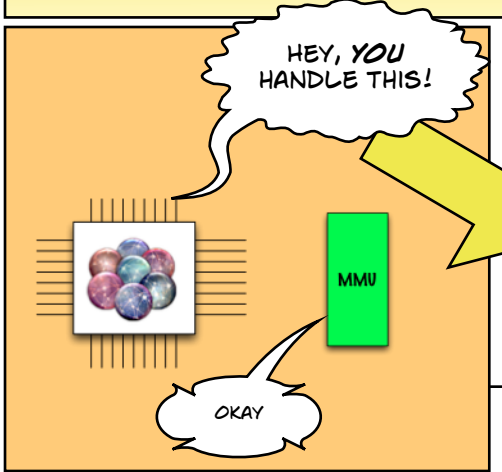


IF YOU'RE A PROGRAMMER, THIS WILL BE **OLD HAT**. PLUS, THESE ARE **GENERIC PRINCIPLES**, NOT **BEST PRACTICES**. BUT YOU MAY NOT KNOW WHAT HAPPENS WHEN THE **NETWORK** OR THE **STORAGE SYSTEMS** GET AHOLD OF YOUR PRECIOUS CARGO. OR **VICE VERSA**.

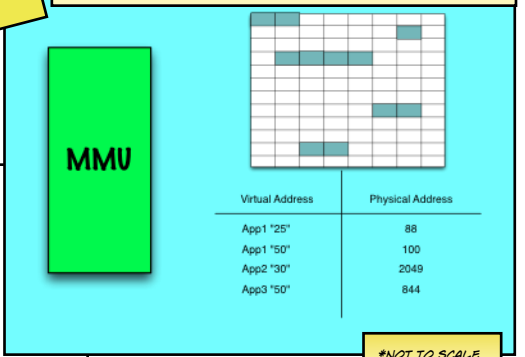
IN OTHER WORDS... **BEAR WITH ME**. :)


EVERY COMPUTER SYSTEM HAS A CPU WITH A BUNCH OF APPLICATIONS RUNNING...
THING IS, APPLICATIONS ON MODERN SYSTEMS THINK THEY HAVE ALL THE AVAILABLE MEMORY.
MINE!
MINE!
MINE!
MINE!
MINE!
!!
... OPERATING SYSTEMS, MAIL SERVERS, DATABASES, WORD PROCESSORS, ETC.

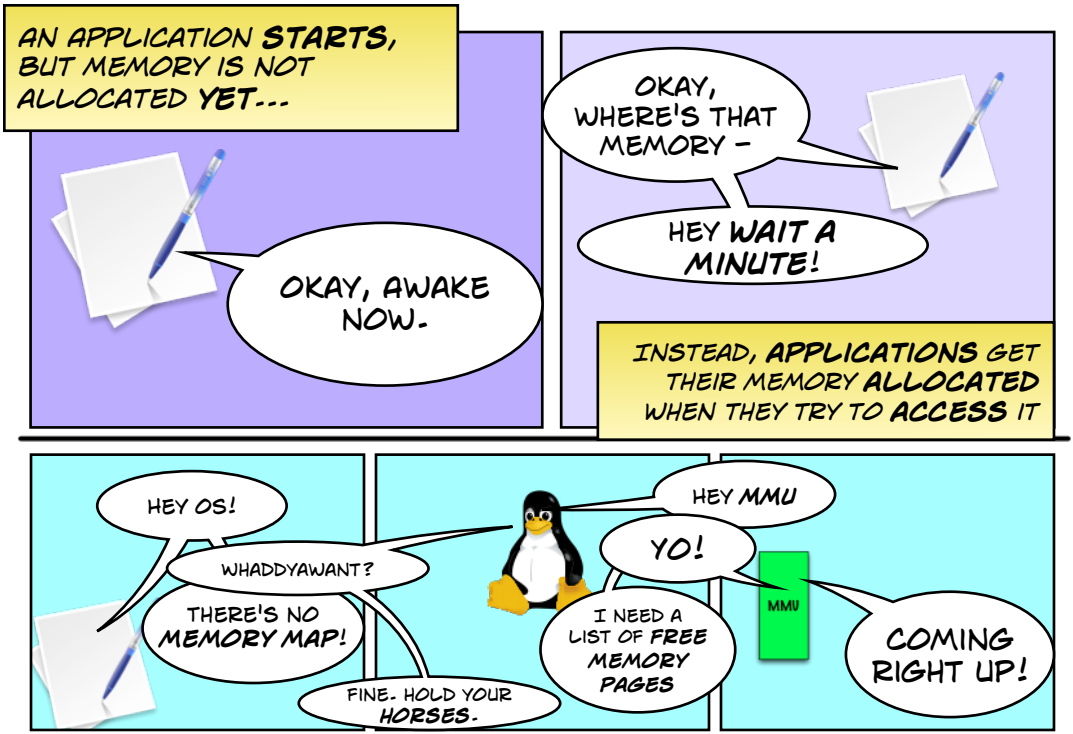
COMPUTE SYSTEMS, THEREFORE, HAVE A **MEMORY MANAGEMENT UNIT (MMU)** TO HANDLE THESE GREEDY APPLICATIONS

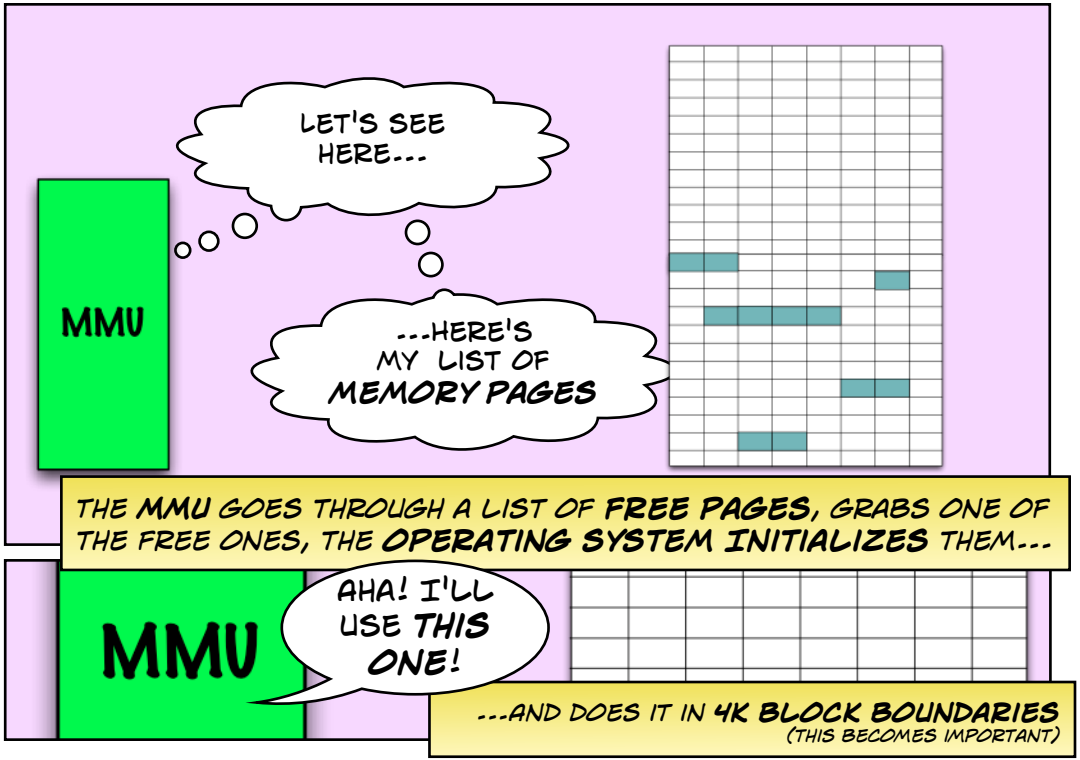


THE **MMU** COMMUNICATES WITH THE **DRAM MEMORY** DIRECTLY AND GETS A LIST* OF **FREE PAGES** AVAILABLE FOR USE



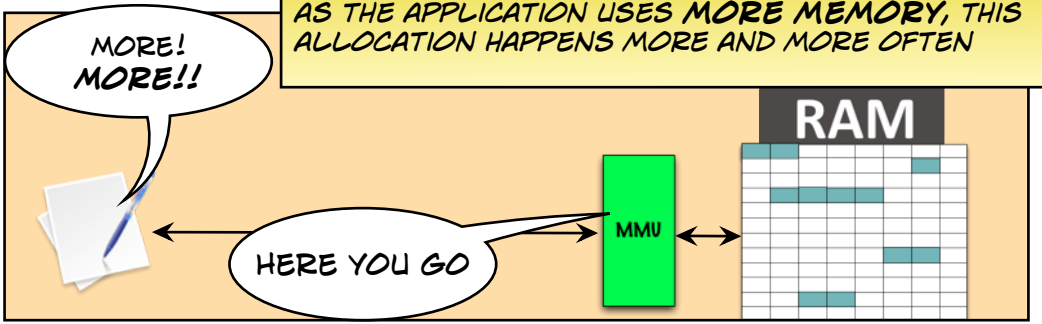
SO LET'S SEE HOW AN **APP** GETS ITS **MEMORY!**


AN APPLICATION STARTS, BUT MEMORY IS NOT ALLOCATED YET...
OKAY, AWAKE NOW.
OKAY, WHERE'S THAT MEMORY –
HEY WAIT A MINUTE!
INSTEAD, APPLICATIONS GET THEIR MEMORY ALLOCATED WHEN THEY TRY TO ACCESS IT
HEY OS!
WHADDYAWANT?
THERE'S NO MEMORY MAP!
FINE. HOLD YOUR HORSES.
HEY MMU
YO!
I NEED A LIST OF FREE MEMORY PAGES
MMU
COMING RIGHT UP!

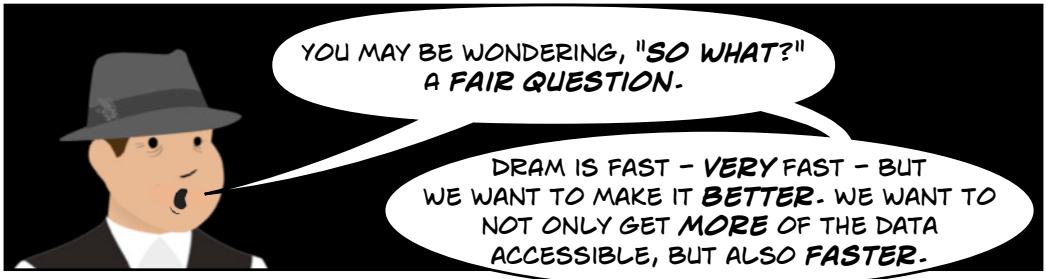

LET'S SEE HERE...
MMU
...HERE'S MY LIST OF MEMORY PAGES
THE MMU GOES THROUGH A LIST OF FREE PAGES, GRABS ONE OF THE FREE ONES, THE OPERATING SYSTEM INITIALIZES THEM...
MMU
AHA! I'LL USE THIS ONE!
...AND DOES IT IN 4K BLOCK BOUNDARIES
(THIS BECOMES IMPORTANT)

THE PAGES CAN BE ANYWHERE IN **PHYSICAL MEMORY**

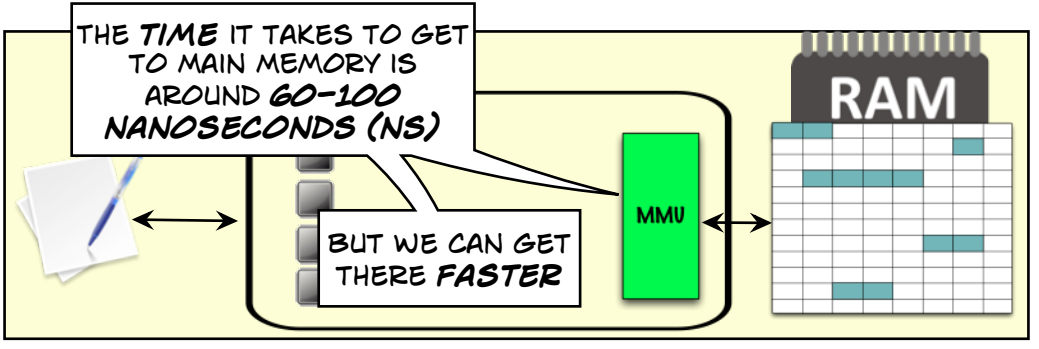
AS THE APPLICATION USES **MORE MEMORY**, THIS ALLOCATION HAPPENS MORE AND MORE OFTEN

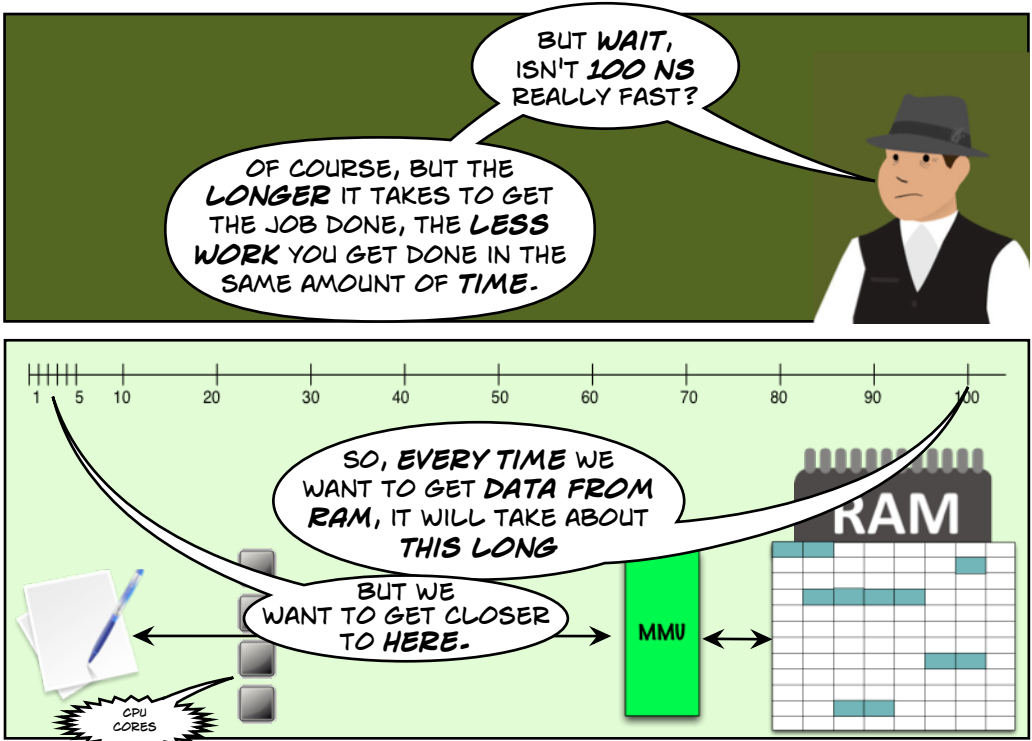


WHEN APPLICATIONS **DON'T NEED** THE MEMORY ANY MORE, THE MMU **RETURNS IT** TO THE FREE LIST FOR THE NEXT APP

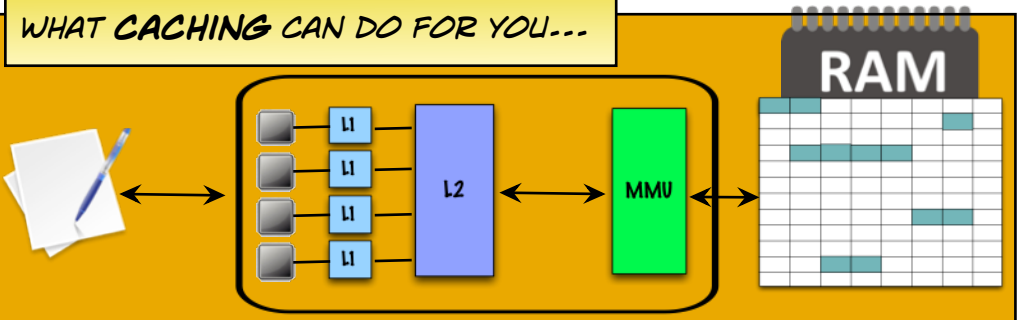


WORKING BACKWARDS...




BUT WAIT, ISN'T 100 NS REALLY FAST?
OF COURSE, BUT THE LONGER IT TAKES TO GET THE JOB DONE, THE LESS WORK YOU GET DONE IN THE SAME AMOUNT OF TIME.
1
5
10
20
30
40
50
60
70
80
90
100
SO, EVERY TIME WE WANT TO GET DATA FROM RAM, IT WILL TAKE ABOUT THIS LONG
RAM
BUT WE WANT TO GET CLOSER TO HERE.
MMU
CPU CORES

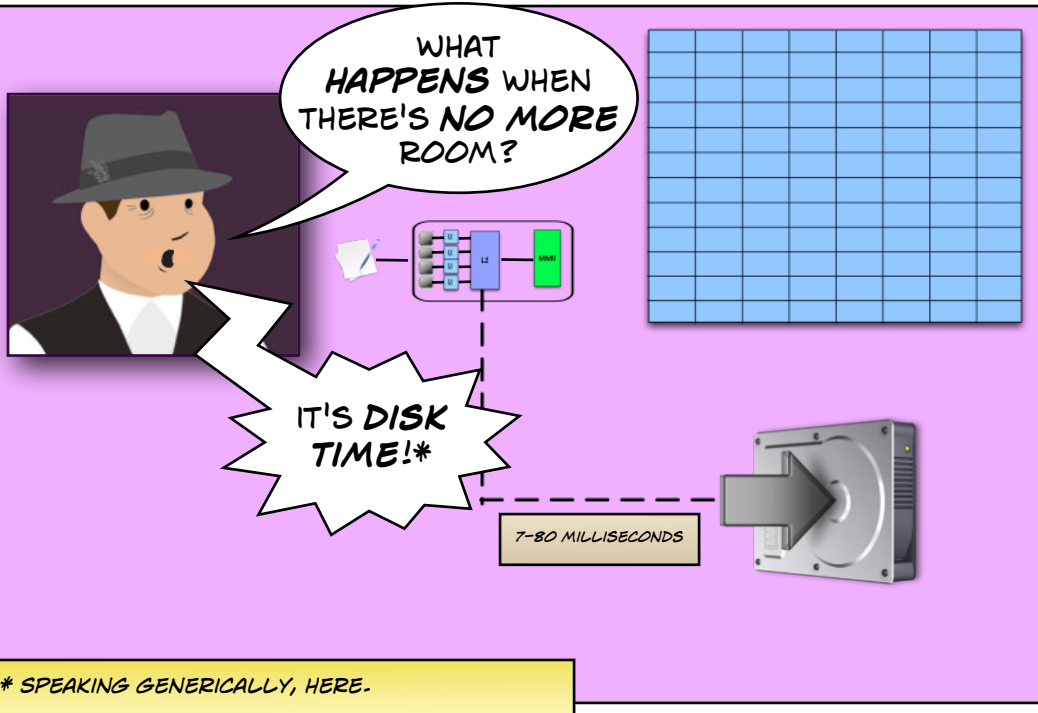
WHAT CACHING CAN DO FOR YOU...



ADDING IN A LEVEL 2 (L2) CACHE BRINGS THAT DOWN TO AROUND **3-6 NS**

A **LEVEL 1** (L1) CACHE, **DIRECTLY CONNECTED** TO THE CPU CORE, TAKES **1 NS** TO RETRIEVE DATA

WHEN AN **APPLICATION** WANTS TO READ FROM **MEMORY**, IT WILL SEE IT'S RUNNING ON "CORE 1" AND **CHECK THE L1 CACHE**. IF IT'S **NOT THERE**, IT CHECKS THE **L2 CACHE**. IF IT'S NOT THERE, IT GOES TO **MAIN MEMORY**


WHAT HAPPENS WHEN THERE'S NO MORE ROOM?
L1
L2
MMU
IT'S DISK TIME!*
7-80 MILLISECONDS
* SPEAKING GENERICALLY, HERE.

FROM A **TIME PERSPECTIVE,** THOUGH, GOING TO DISK IS **EXPENSIVE.**

THINK OF IT THIS WAY - IMAGINE YOU WANT TO HAVE PIZZA DELIVERED TO YOUR HOUSE, AND THE **L1 PIZZA** PLACE IS **1 KM** AWAY.

IF THAT PIZZA JOINT DOESN'T HAVE YOUR PIZZA, THE NEXT ONE (**L2 CACHE**) IS BETWEEN **3-6 KM** AWAY. STILL DOABLE, BUT YOU'RE GONNA HAVE TO **WAIT** A LITTLE LONGER. (A **PAIN** IF YOU'RE HUNGRY!)

IF THAT L2 PIZZA JOINT **DOESN'T HAVE** THE PIZZA YOU WANT, THE NEXT ONE IS **100 KM** AWAY. (HOW LONG DOES IT TAKE TO DRIVE 100KM?)

IF **THAT** PIZZA JOINT DOESN'T HAVE YOUR PIZZA, THE **NEXT** ONE - A DISK, IN THIS CASE - CAN BE UP TO **8,000,000** KM AWAY.

GREAT, I'M COLD NOW

OR **MORE THAN TEN** ROUND TRIPS TO THE **MOON!**

YES, I KNOW I SWAPPED THE TIME/DISTANCE FOR THE PURPOSE OF THE ANALOGY. WORK WITH ME, HERE!

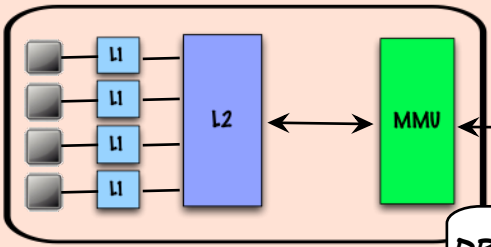
TWO THINGS BECOME IMPORTANT TO NOTE:

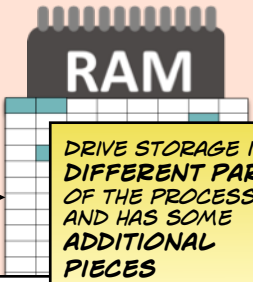
FIRST, WHENEVER YOU CAN, YOU WANT TO KEEP THE STORAGE **AS CLOSE** TO THE APPLICATION AS **POSSIBLE AND REASONABLE**

SECOND, NOT ALL **USE CASES** ARE APPLICABLE FOR ALL KINDS OF **STORAGE ENVIRONMENTS**. AFTER ALL, WHO WANTS **COLD PIZZA**?

THE **DIFFERENT TYPES** OF STORAGE CAN MAKE A BIG **DIFFERENCE** FOR **PERFORMANCE**, AS WELL AS **TYPES OF WORKLOADS** (E.G., THE PIZZA DELIVERY PROBLEM)

USING A STORAGE DEVICE





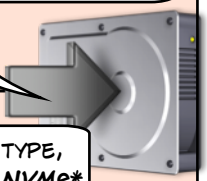
DRIVE STORAGE IS A **DIFFERENT PART** OF THE PROCESS, AND HAS SOME **ADDITIONAL PIECES**

I/O

DRIVE/DISK TYPE, LIKE SPINNING **HARD DRIVES** (HDDs) OR **SOLID STATE** (SSDs)/**FLASH** DRIVES

I/O MMU ALLOWS THE APPLICATION TO **ACCESS MEMORY** ON THE DRIVE **WITHOUT NEEDING** TO GO TO THE CPU FIRST

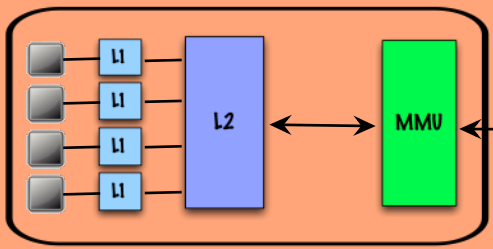
ADAPTER INTERFACE, USUALLY **SAS, SATA**, OR **PCIe***

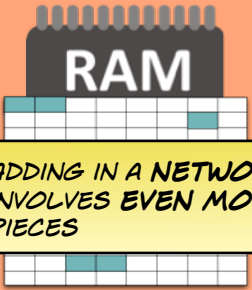


PROTOCOL TYPE, LIKE **SCSI** OR **NVMe***

* MORE ON THESE LATER

NETWORKING A STORAGE DEVICE





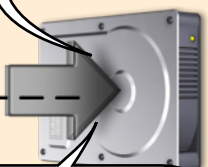
ADDING IN A **NETWORK** INVOLVES **EVEN MORE** PIECES

SERVER INTERNALS

I/O

DRIVE/DISK

ADAPTER INTERFACE, LIKE A **HBA**, **HCA** OR A REGULAR ETHERNET **NIC***



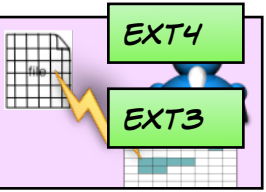
STORAGE **NETWORK TYPE**, LIKE **FIBRE CHANNEL**, **FCoE**, **INFINIBAND**, **ISCSI**, **NFS**, **SMB***

DRIVE/DISK TYPE

NETWORKING A DRIVE

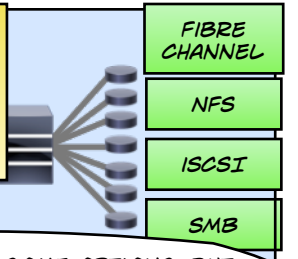
* MORE ON THESE LATER

IN **PART 2**, WE'LL TAKE A LOOK AT **FILE SYSTEMS**, AND THEIR ROLE IN **HOW DATA IS STORED**.



IN **PART 3**, WE'LL TAKE A CLOSER LOOK AT HOW **DRIVES** (BOTH **DISK AND FLASH**) HANDLE **DATA STORAGE**, AND WHY THAT CAN BE **VERY IMPORTANT**.

THEN, IN **PART 4**, WE'LL LOOK AT WHAT HAPPENS WHEN YOU **MOVE** THE DRIVE **FURTHER AWAY** FROM THE **CPU**, AND HOW **VARIOUS NETWORKS** CAN MAKE A **DIFFERENCE**.



VERY SPECIAL THANKS GO TO **JOE PELISSIER**, DISTINGUISHED ENGINEER AT **CISCO**, FROM WHOM THE TECHNICAL CONTENT CAME.

@DRJMETZ ON TWITTER

AFTER THAT, WE'VE GOT SOME OPTIONS, BUT EVERYONE LIKES A LITTLE **MYSTERY**!

TO BE CONTINUED...