proof using contrapositive.

PROJECT (ECS)

9.2.2020

- cpu vs gpu (start).
- gpus - simd (in lock step, velocity local)
- cpu's - mimd how manage close parallelism)
- how they use far parallelism with cores / multithreads.
- what is loose coupling and what is tight coupling.
- it depends upon the code that uses it.
- entity ——— all variable are used directly in update(), so definitely its tightly coupled.
- but if we are processing the data separately then no more it is any tightly coupled.
- component based data now becomes tightly coupled.
- in essence, the component - systems arch. respects both data cache and code cache. -
- isn't this like gpu already? they do the vertex shader first and then fragment shader. (other shaders). — basically (lock-step (ex.) and.
- they love purity. (very)
- a shared common interface among very implem. (like in Unity NativeArray)
- how can compiler optimize, should you tell it? (Temp. TempJob)
- struct <as is>, struct <A> official., <A> <?>
- how to specify which implementation (A, A, A, B)?
- ~~how~~ should coc assume purity?
- how to check if side effects a = sort(a).
- how matrix is structured / dense or sparse?
- gcc how structure padded / aligned attribute.

- gpu's work with small chunks of data at once (from GPU ram). called bins. and one command processor commands. all the units to work.

- gpu's also like cpu alu, they want to utilize their adders and multis. one. dividers + special funcs. (math). in parallel. with help of vliw instructions.

- since gpu's dont have branching, vliw is a nice way to run the cores, because they are simd.

- lock-step means no need of too many synchronization wires, and too many command cores.

- why is GPU not the prevalent CPU architecture?
  CPU + embedded processors + GPU
  (unpred.) (pred) prev. (simd).
  (no cache, no pred)

- Serialized data access is performant.

- how can data structures be designed for serial access. if you need some field next, why not let the field be the next in the structure (depends upon the various algorithms using it, which is why profiling is important).

- amdahl's law based optimization by the compiler itself.
- how should various fundamental data structures be designed such that they are performant for CPUs / GPUs.
- what is the benefit of GPU sharing the same instr. set with CPUs (RISC-V)?
- is it possible to separate out the optimation logic / synthesis from the programmer code. Can the compiler do automatic profiling & ML for the appropriate implentation.
- how high-level algorithms can be written.
- is there a way to quickly choose appropriate implementation based on constraint (p* -> p)? (comp.)
- maybe using a suggestion from programmer?
- moving towards functional design - helping in separating out logic from data (plain old data).
- how complex the CPU pipeline is, what a CPU ~~stall~~ pipeline stall is? structural, data, control.

- how can system be made more concurrent, more decoupled?
  (spatial OS).
- impbrobale is using the concorrency of ECS architecture to simulate large worlds.

- how user input is given to server, which gives back authoritative state of player and how the manage packet loss (rate↑) and high latency (prediction).

- how can we minimize latency at every stage of processing. processing physics at various LODs.

- how can we store the information of a space (map area) state in a compressed way and then steam on more accurate information (kind of like jpeg). (image sharpening).

- how can a map location information be maintained
- how
- how can super large physics effects. (effects that need a large no. of components to come together work).

{ satellite looking at night sky }
{ LOD - compute in local blocks }

testing performance of data structures for ECS freiendly - how performance with perf-stat.

why should OS be written in C, if we want OS to execute various algorithms why do we want to use high level languages to implement OS.

this can make designing OS beginner friendly again (through decoupling)? libraries for making an OS. kind of like RISC V. ~~ang~~ - open source ~~inter~~ SW interface for application (the internal OS design can be customised).

no more android; windows / only the SIA will matter.

how can this SIA be optimized for high concurrency, high data throughputs can

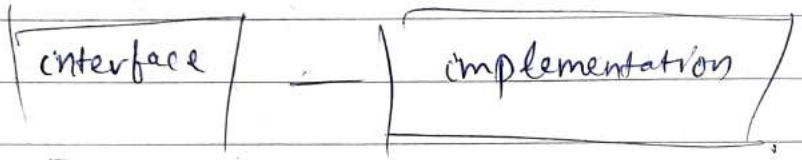
can an interface's throughput / latency on a given architecture be predicted, given a use case.

then, can given a groupp of similar use cases, the SIA be auto-designed (through some kind of space search?)

can then an appropriate implementation be packed together by compiler.?

can an OS present multiple SIA based on application usage / request. think of every application requesting an SIA kind of like OpenGL / Vulkan / DirectX.

can the same thing be done for processors.



is more indirections always solve problem? when is it too much?

is functional decoupling better.?

localized side effects always better, all in one place - overwatch.

which is the higher layer and can do without perf. optimization? this keeps on changing as we try to solve larger and larger problems.

netcode - why is this hard. they expect loss - it involves unpredictability. how can we exploit client-cpu power if available?

how to distribute loads for better concurrency (remember MIMD systems). how to minimize communication overhead. how to share low LOD information with low latency. how to minimize latency to screen. this is very critical for VR/AR.

Side quest: can program synthesis / program proof be done for crypto currency?)

(soln.). space search is critical for crypto currency?.

mutation based AI Synthesis, could it be a candidate for space search. Randomi

randomize NP algorithms suitable for crypto currency workloads.

so they are also suitable for concurrency because - low data high space search. (complex).

NP hard, what is it?
can it be used for concurrency.

such problems in game design?

data intensive" — or space search intensive)

then how do par concurrent programs synchronize.

~~Unity~~ ECS
- burst compiler
- C# job system

optimized native code using LLVM

Stand on the shoulders of giants.

no gc (or custom)
custom memalloc.

can components be made separately and shared among people (decoupling).

can we get program overview (insight) (without reading it) at different levels.
some kind of plugin / website / startups.?.

(just insight like headline insight on food combined from research papers).

github is already doing it with language detection. we can count ifs, loops, variable in functions, dependencies, dependants ... (security). (performance for a given arch.).

can an game / app. dynamically pick up an implementation based upon the arch. of machine it is currently running on / the software already available on the machine.

in CO packages implement X interface either can we dynamically choose any one of them based on what is currently available on the system. may solve the yet another plugin problem.

also about versioning. interface more important than versions. as long as version remains same, its ok.

how can transition systems be used for this (maybe this could be a better way to define algorithms).

each level of transition system is an interface (think actions). there is a hierarchy of transition systems.

tree of transition systems. using it to model concurrent sys

bitonic sort / random sort.

↳ as you can see, each sort has the same interface, but a different implementation.

what one arch. needs is not necessary by others
→ disk defragmentation
→ array sorting -
→ precomputed buffer.

each architecture has its own needs,
how can a compiler insert array sort before loop, without changing code behaviour, (but improving speed).

double / triple buffering for computing physics objects instead of copy can help avoid data hazards (partially modified buffers).

intel core, intel pentium, AMD bulldozer pipeline stages.

ralf brown interrupt list.
(interface description on regs) → this changes!

interface - description on music (rock) (cool/bad)
(Click Youtube) (more insights - better?!) (boy/girls singing)
(interfaces on music instruments!), (implementations)

ex- of common interface different implementation.
SPS drill bit, screw bit, clamp bit.

OSB.

promises. Promis.all() requires sync. basically like "barrier" (job) in burst compiler. but its always better to minimize synchronization.

how long can we delay syncronization.

If u an text processing 200 different texts, why do i need to wait till all of the 200 texts are available?. but many js. libs do that (i do that too).?

it would be cool if promises could be handled transparently by ~~higher~~ for-each higher level logic. automatic handling of coroutines just like nodejs event-loop.

when to do lazy processing and when to do it now. haskell approach.
lazy execution model V/s. eager execution model.
does this have something to do with CPU cache size.

similarly, when is it better to have linked lists, instead of contiguous arrays.?
yes. in some cases, linked lists are more performant
how can this idea be extended to trees, graphs.

iceberg based interface, bottom can change because you dont see it. so for ship. person carrying shard or bomb? the look of buildings.? outside interface same, different from inside.

when is homogenity bad. when should interface change based on use case?

when to use einstein's relativistic physics vs when to use newtonian physics.

moto gp. when to use skeleton renderer, when to use rough renderer, when fully textured rendered.

game - scrap mechanic, creativerse - lots of actions, items. how they interact.

collision detection - physics - explosion simulation.

cloth simulation, LOD. when more from far, but accurate from near, more users => more realistic simulation. tree falls in jungle (some one to see). how to do persistent maps.

space trash, someone keeps simulating it disaster. => necessary in space games => more players

function tags, just like package tags, repo tags, data structure tags. interface tags. implementation tags.

could be useful for protein simulation, in cells., body subsystems games. life to cells, operation of organelles. games at different scale., a bug's life. how life perspective changes with scale.

you will get 0 this semester if you dont make a game need pressure, but i also need. ~~motivation~~ to be interested in it.

also like the concept of edge of tomorrow, that movie where the brain in end., ~~game~~. games., resume exactly at a point and try again (in multiplayer games.). how can it be done? dota. apex legends. checkpoints in multiplayer games. (using AI?) respawn? ie. maintain as player godlike?

replaying full match as with AI + checkpoints.? think experiment = fun?

game interface, separate implementation! Why pay for developers?

how to do away with authoritative server, and make a fully distributed networked game!

is uniform interface always better.
SSD (flash) vs intel optane.
(DSA). - everything will not autoboost.

question remains, how to parallelize inherently serial operations. what is truly inherently serial. can we get some numbers. (max min range).

how to do predictable execution. (when necessary?). when how can it be implemented. when is it an implementation and when an interface (flag).

{choosing a particular interface}.

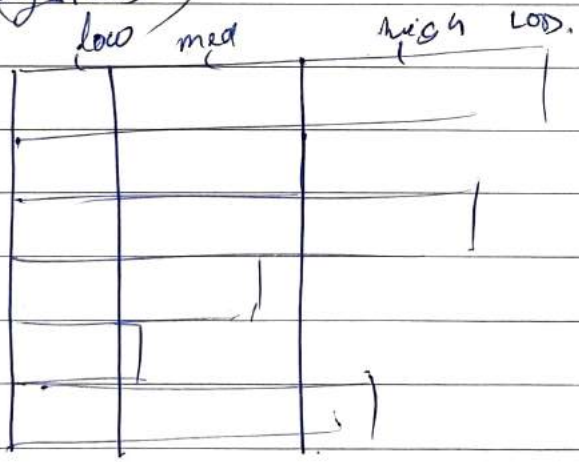
distributed data storage - DS class.
coding scheme - useful for large maps. (ex). (persistent).

system using components. singletons, tuples. archetype. : more constraints => better usually? hole of success. (depends on each.?)

data oriented technology stack (DOTS) ECS.

entity	component	systems
id.	physical	logic 1
	speedcomp.	logic 2.
	health.	physics
	inventory.	ai
		game ← (inventory)

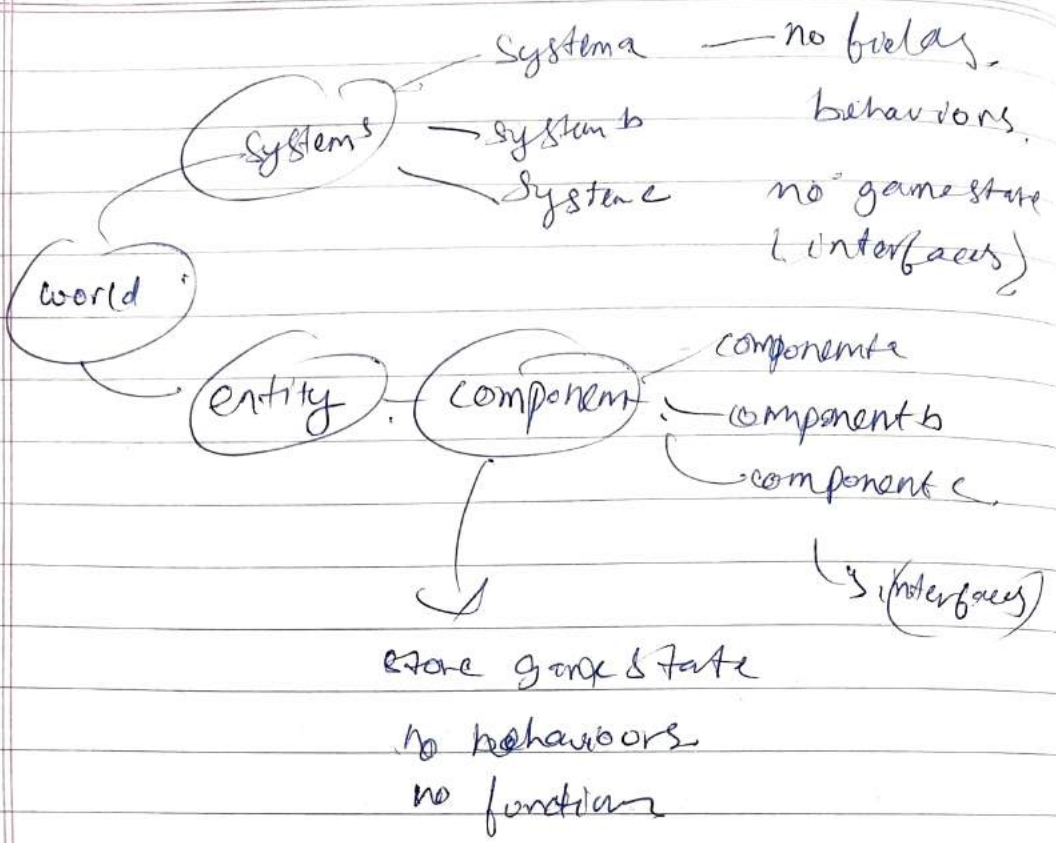
how binning work?
what is level of detail (LOD)?
how can it be suitable for physics components?
(systems)



1990s actor model.

2000s. component model (game object?)

manage complexity on a quickly growing codebase
visitor pattern? functional types pattern?



Systems only care about what components they operate on.

Systems can be added with ease.

behaviours single call site,

pure functions.
reducing side effects.
preferrable in a angle area

singletons deformance.