SaarCOR – A Hardware Architecture for Ray Tracing

Jörg Schmittler,
Ingo Wald, Philipp Slusallek

Computer Graphics Group,
Saarland University, Germany

Why Ray Tracing?

We already have rasterization!

Ray tracing …

- Provides physical correct images by default
  → no need to approximate effects
- Scales logarithmically with scene complexity
- Has built-in occlusion and frustum culling

Previous Work

Interactive ray tracing on various platforms:

- Supercomputers
- Clusters of PCs [Wald'01]

→ it scales well in number of clients

- GPUs [Purcell'02] and streaming processors

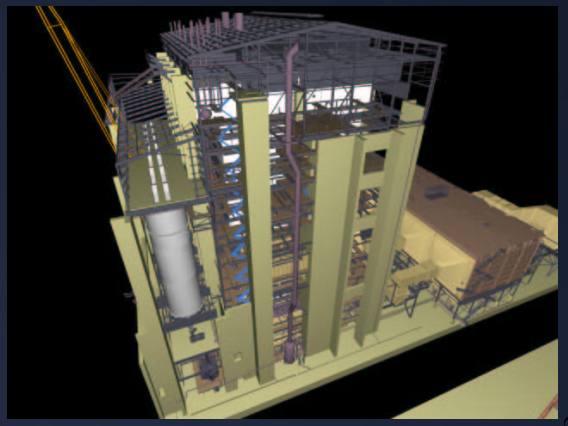
→ promising, but not yet efficient

- Special purpose hardware

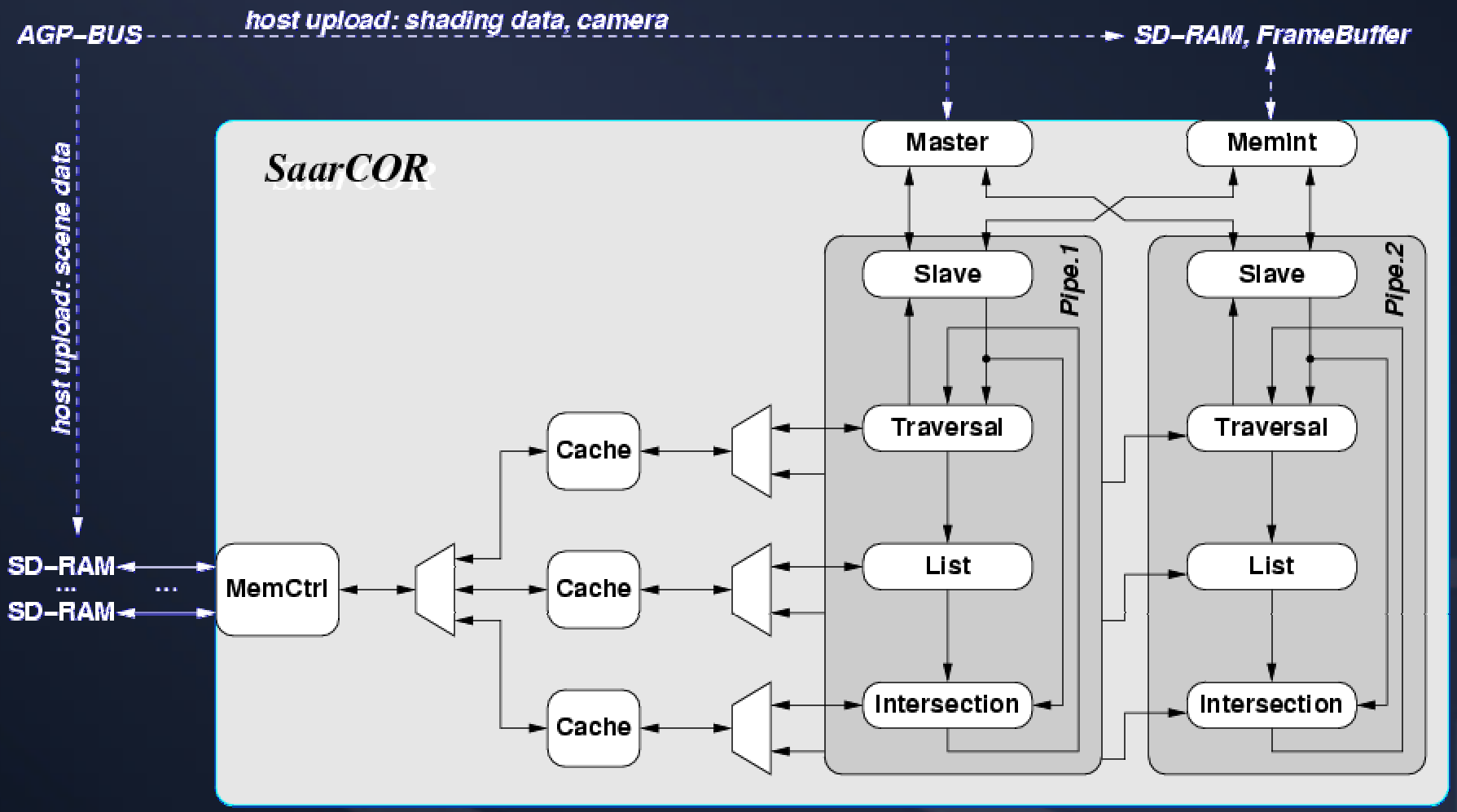
→ estimate the lowest cost for ray tracing

SaarCOR Architecture

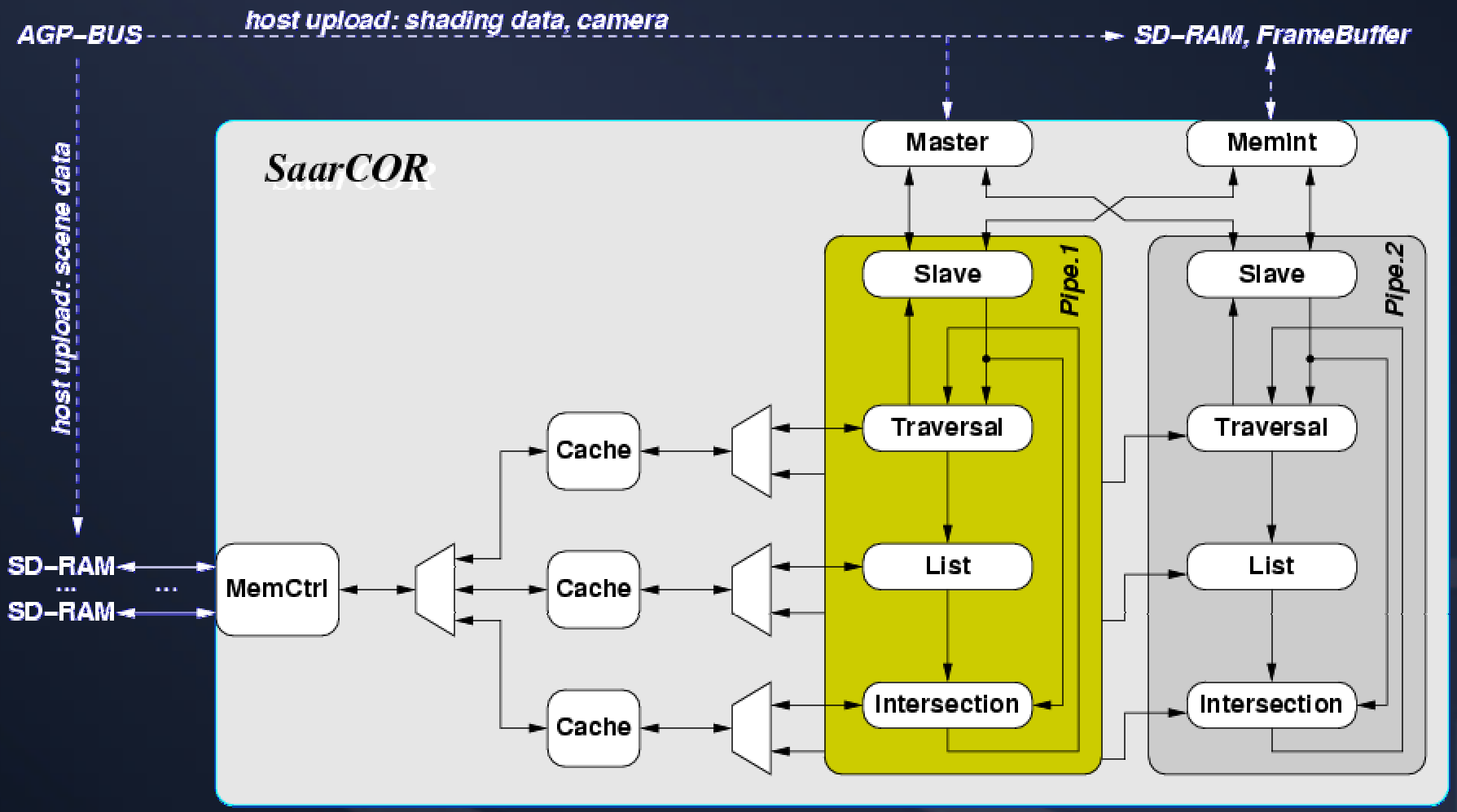
- Based on interactive software ray tracer [Wald'01]
- KD-trees as accelleration structure
- Pakets of rays to reduce bandwidth
- Fixed OpenGL-like shading
- Additionally shadow and reflection rays
- Simple low bandwidth memory interface
- Half the floating point performance of GeForce3
- Achieves frame rates comparable to today's gfxcards



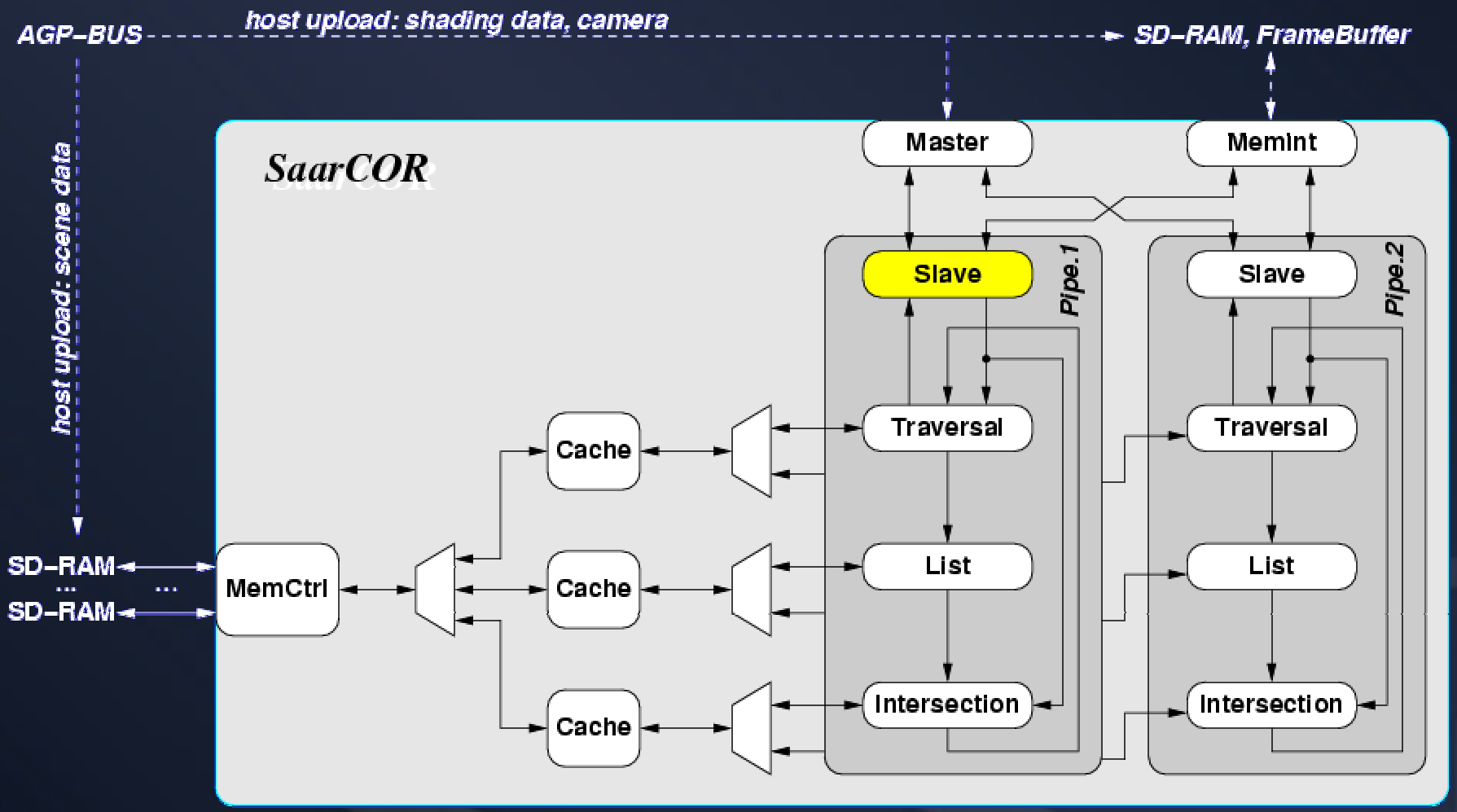
SaarCOR Architecture:
System overview



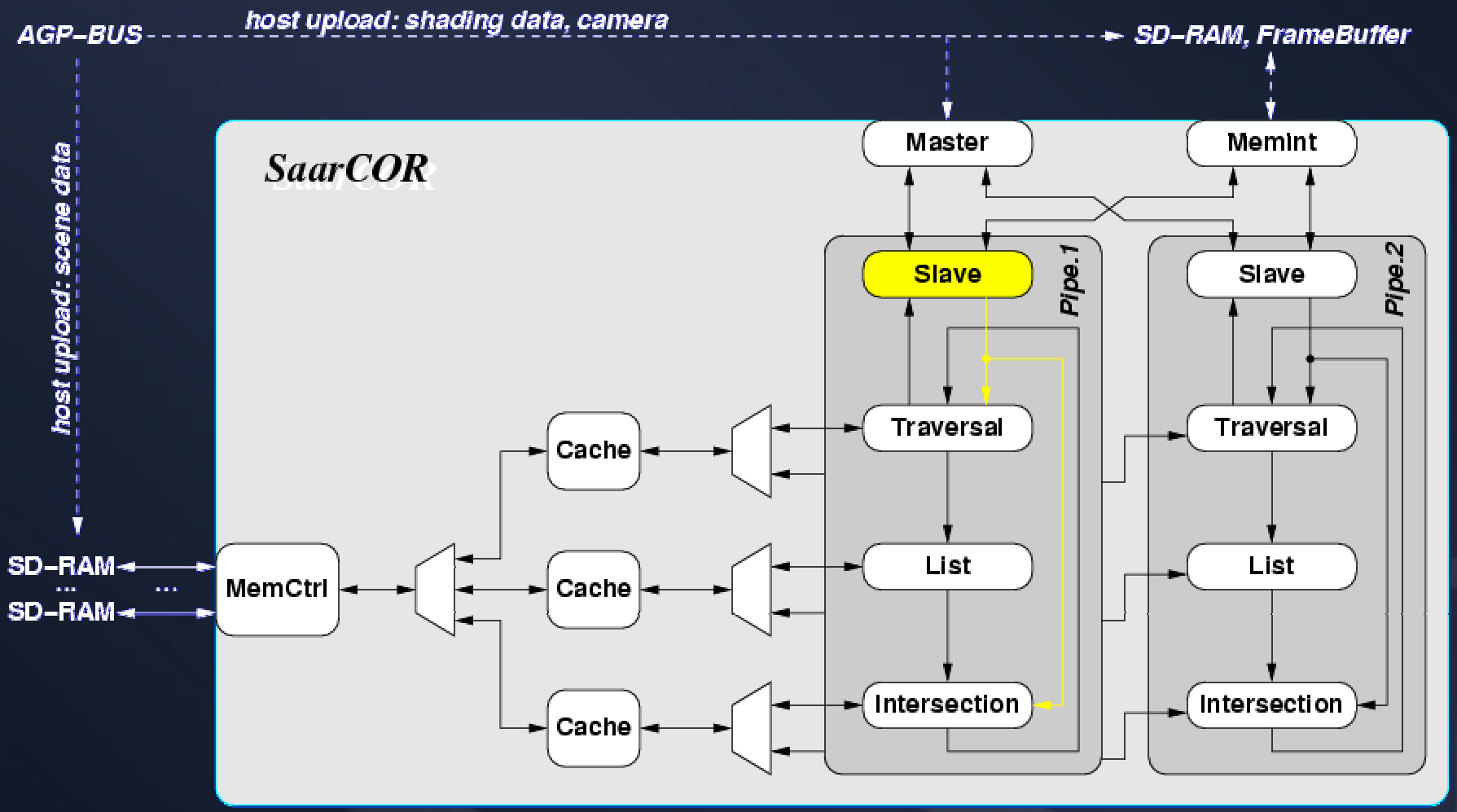
SaarCOR Architecture:
System overview



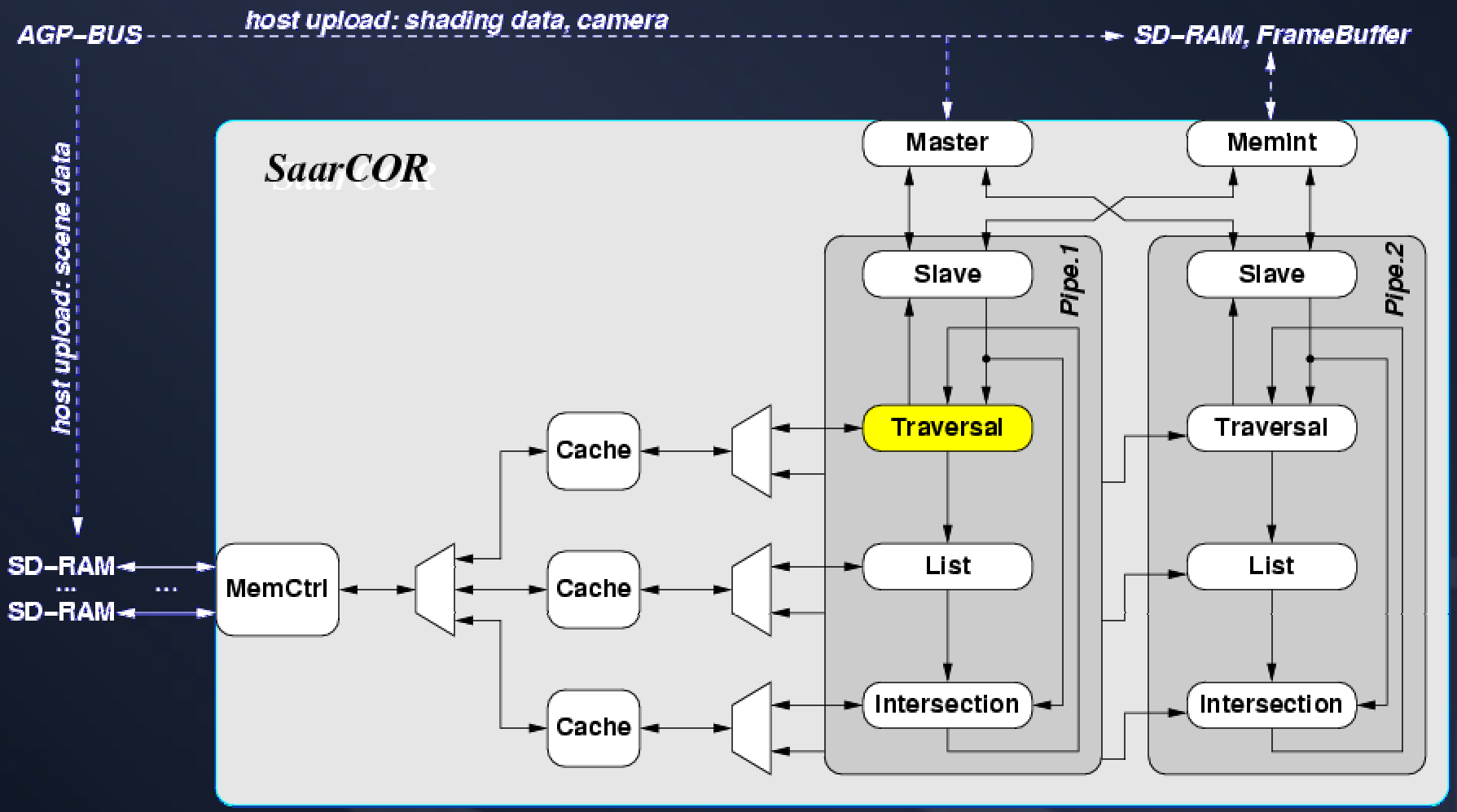
SaarCOR Architecture:
System overview



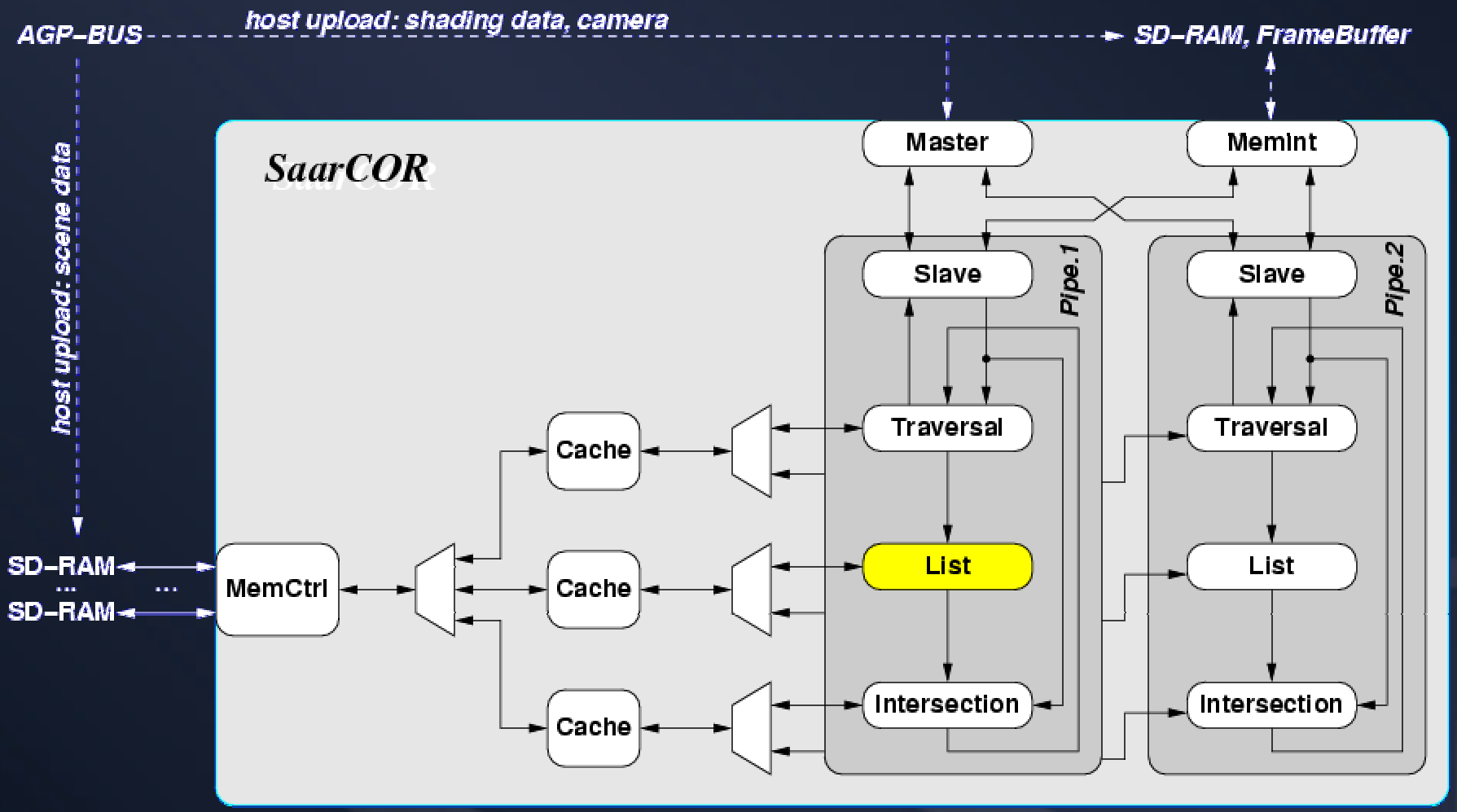
SaarCOR Architecture:
System overview



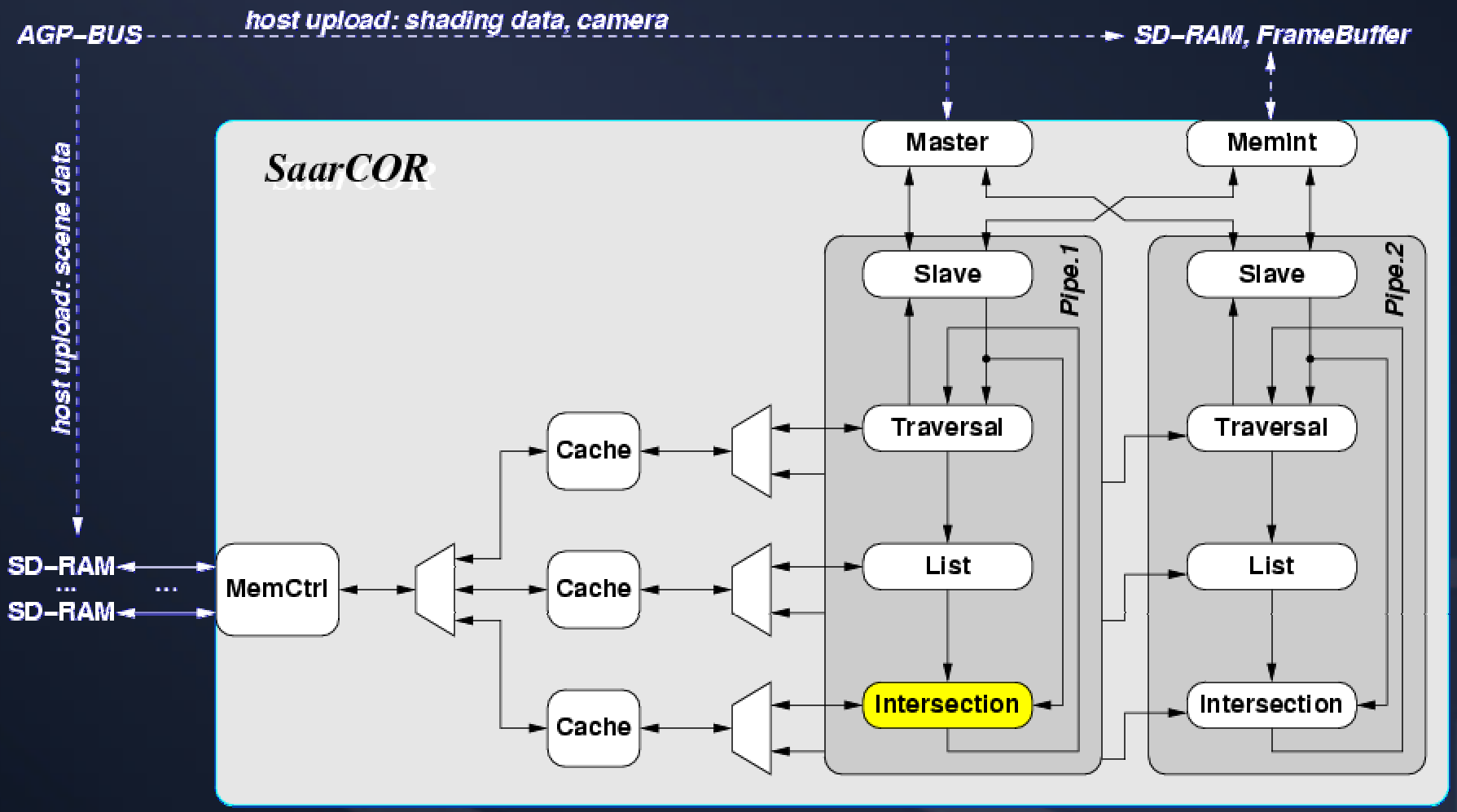
SaarCOR Architecture:
System overview



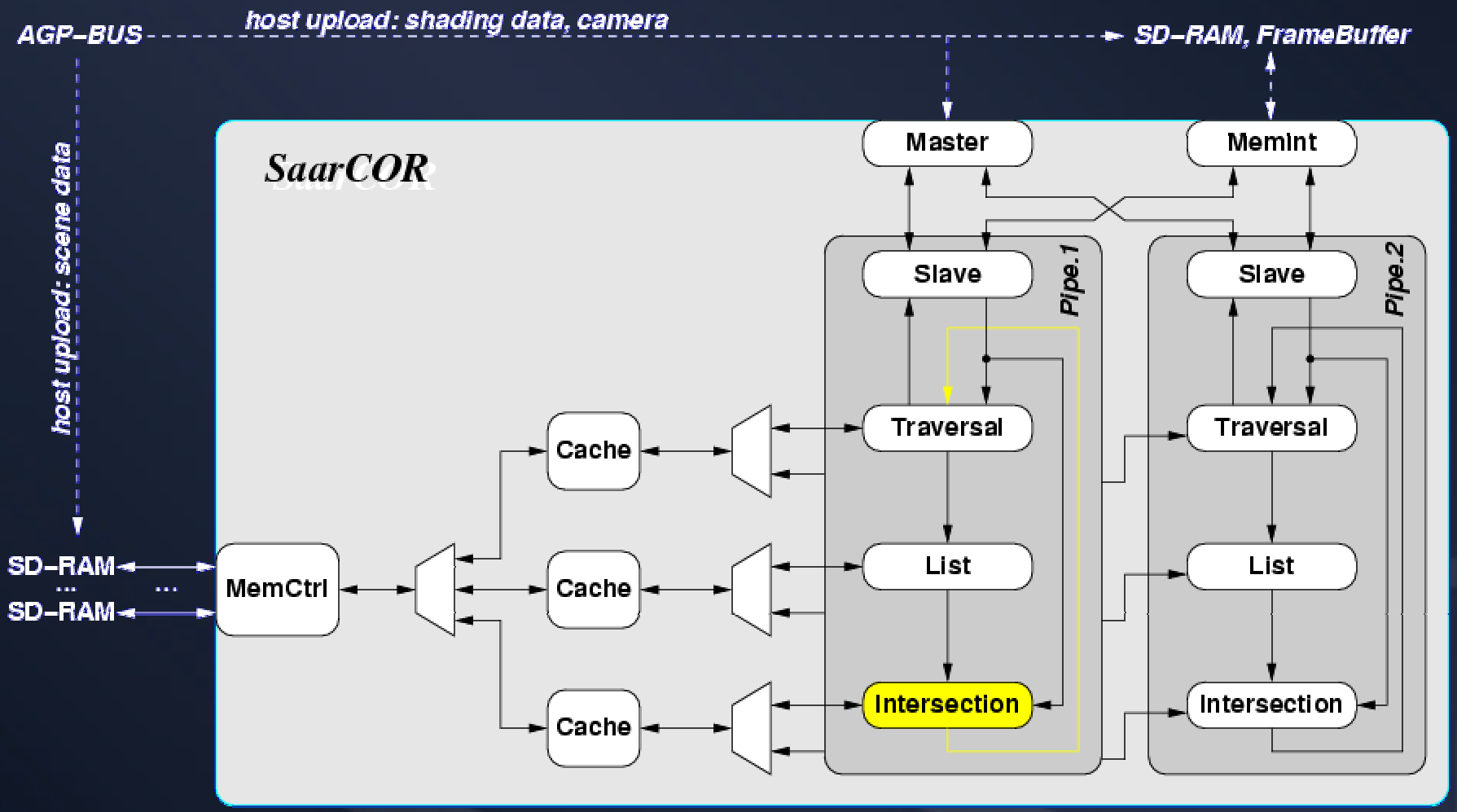
SaarCOR Architecture:
System overview



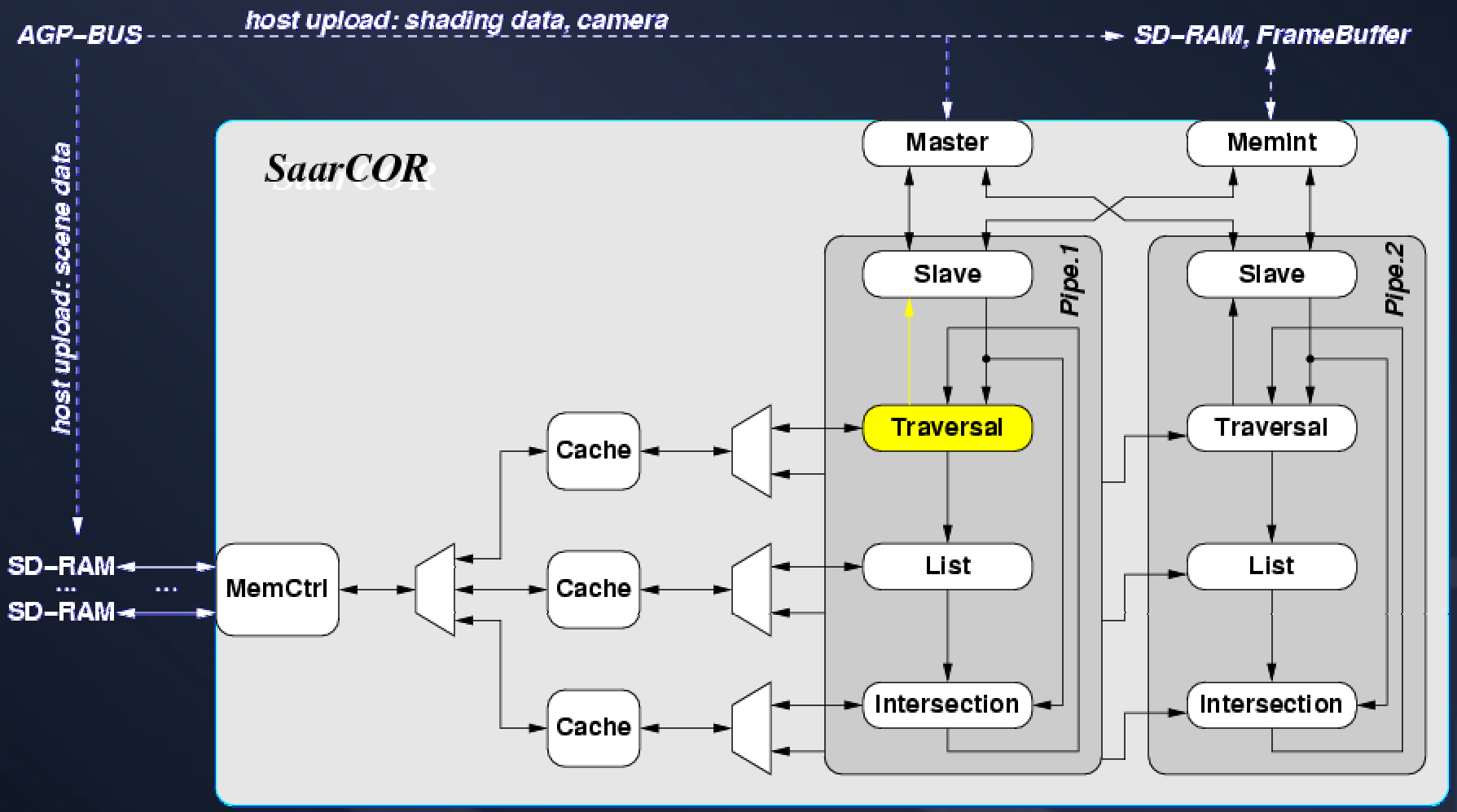
SaarCOR Architecture:
System overview



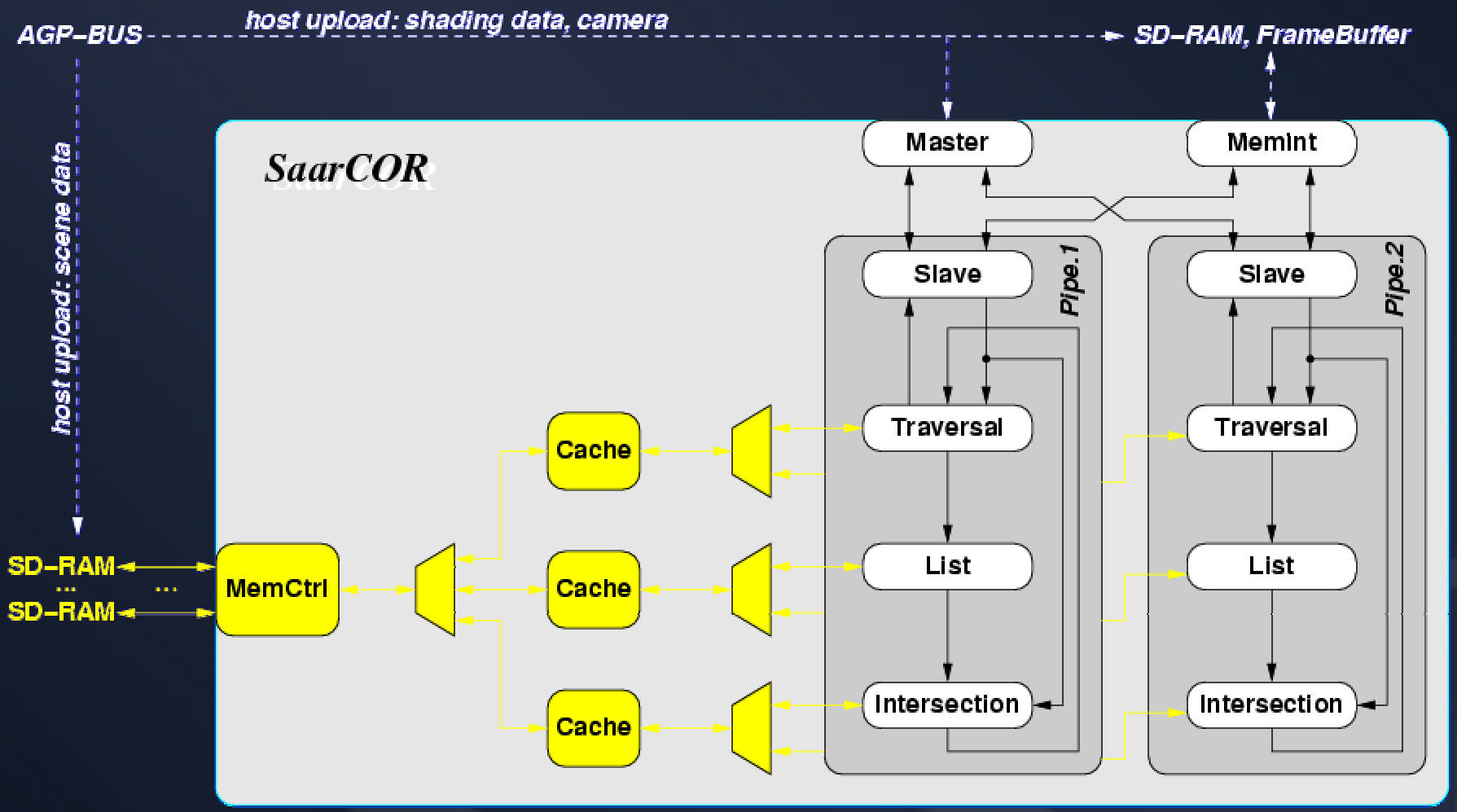
SaarCOR Architecture:
System overview



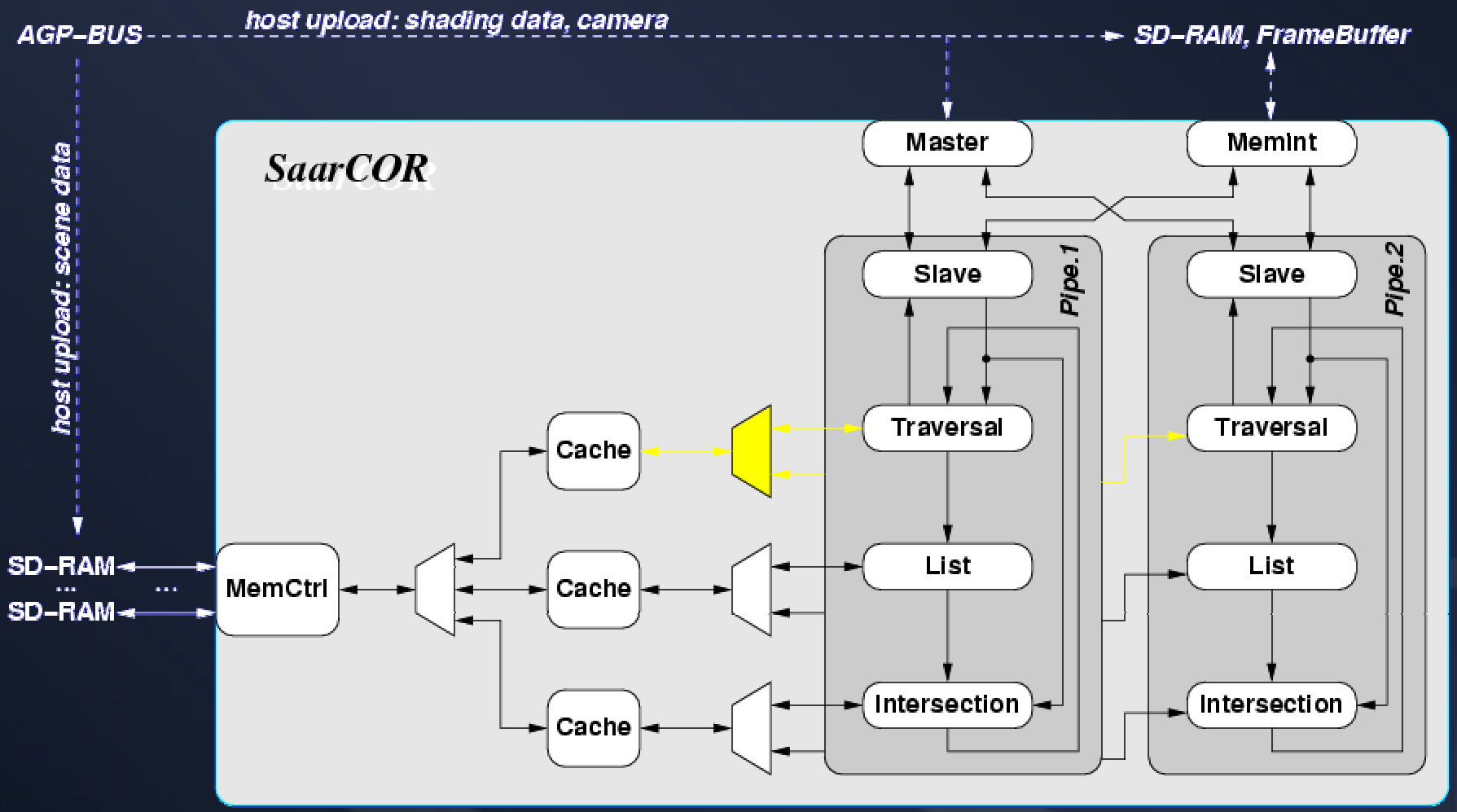
SaarCOR Architecture:
System overview



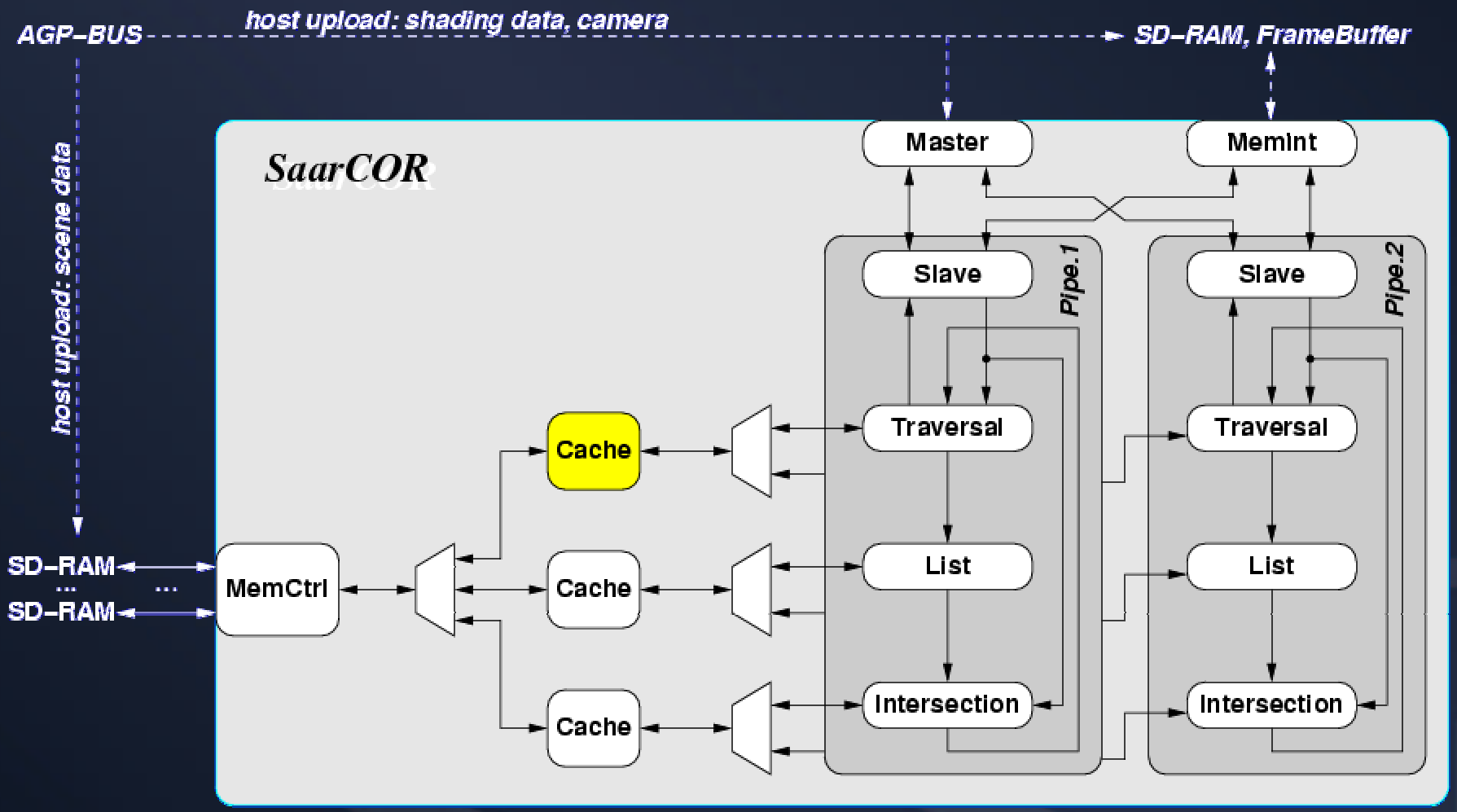
SaarCOR Architecture:
System overview



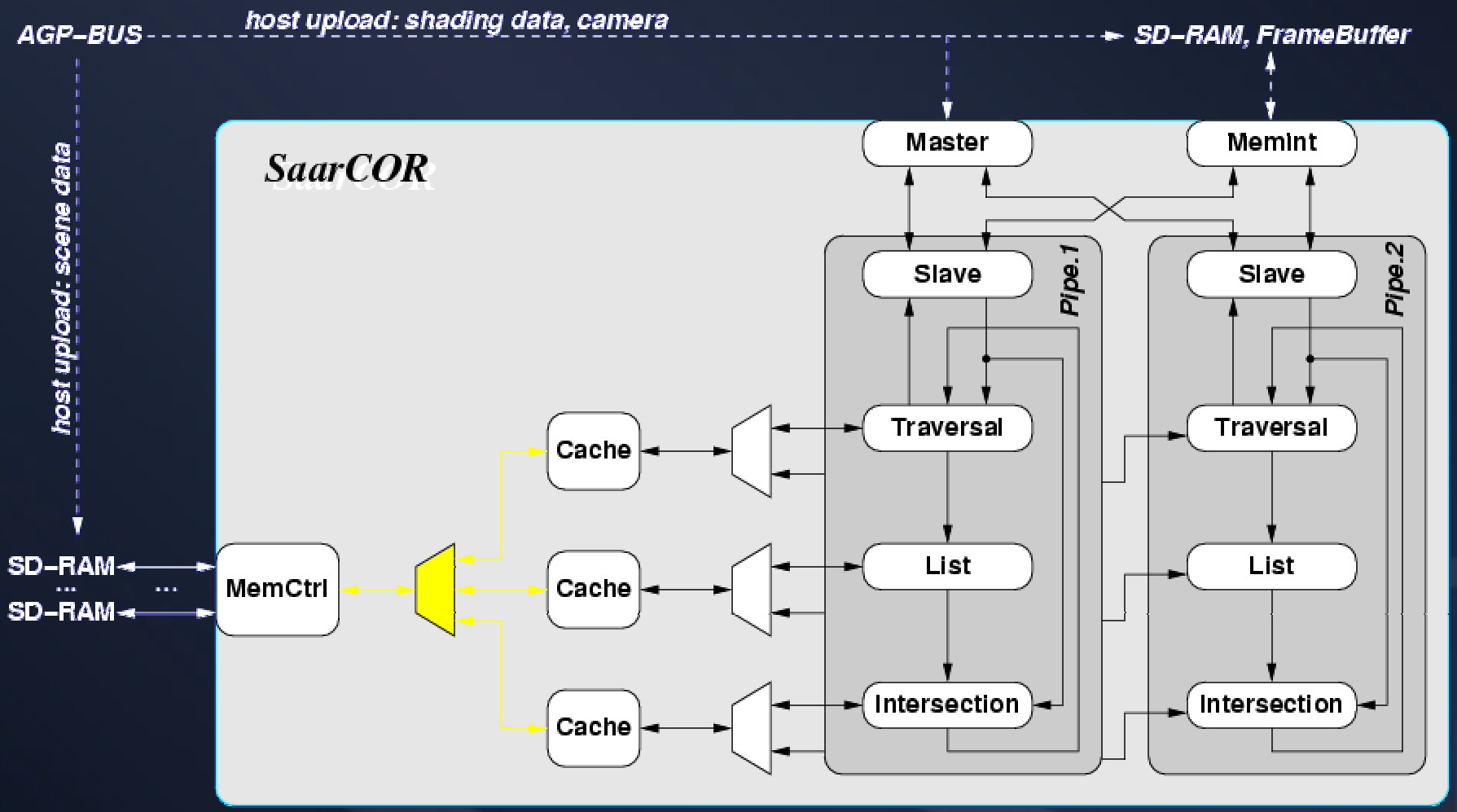
SaarCOR Architecture:
System overview



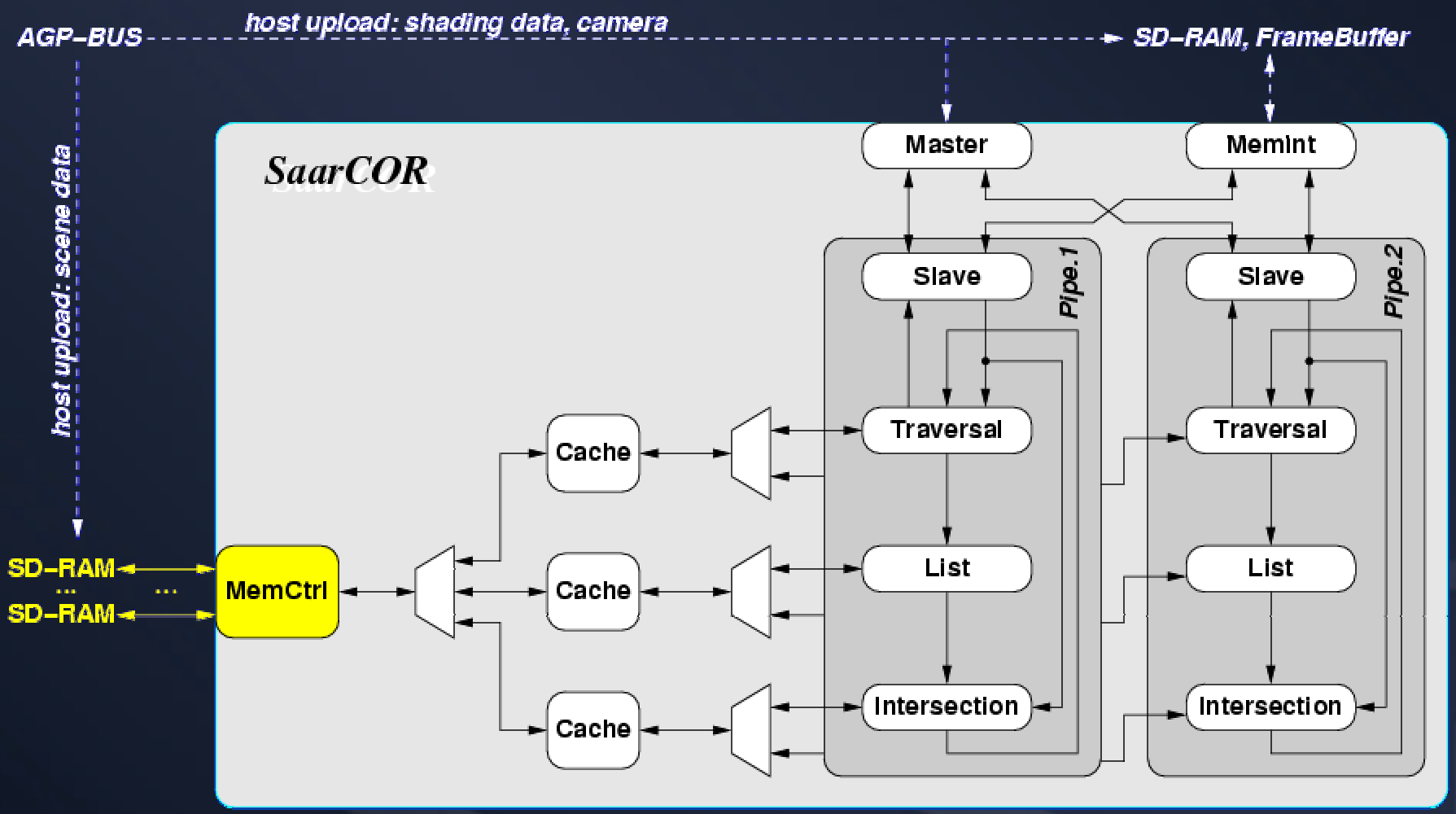
SaarCOR Architecture:
System overview



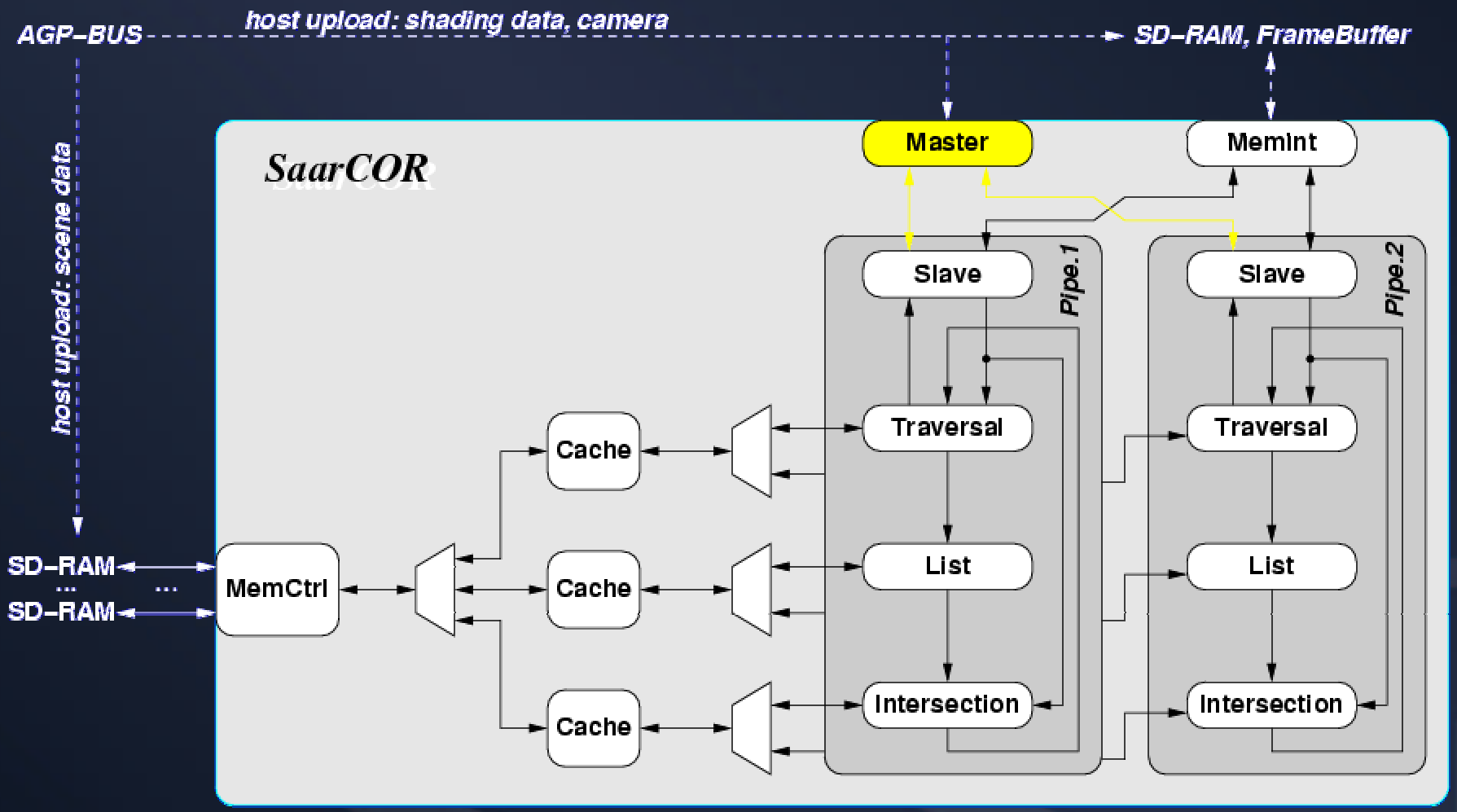
SaarCOR Architecture:
System overview



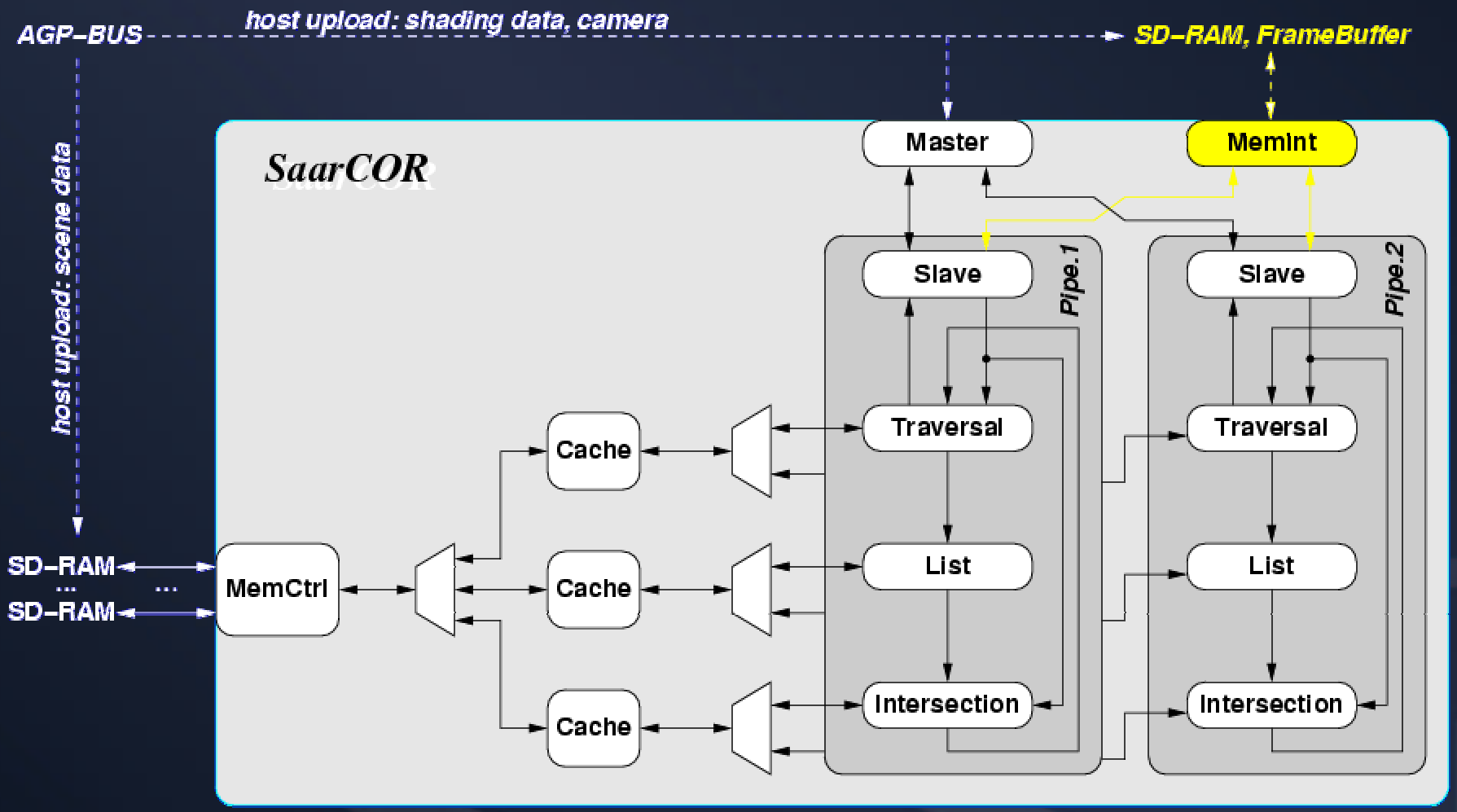
SaarCOR Architecture:
System overview



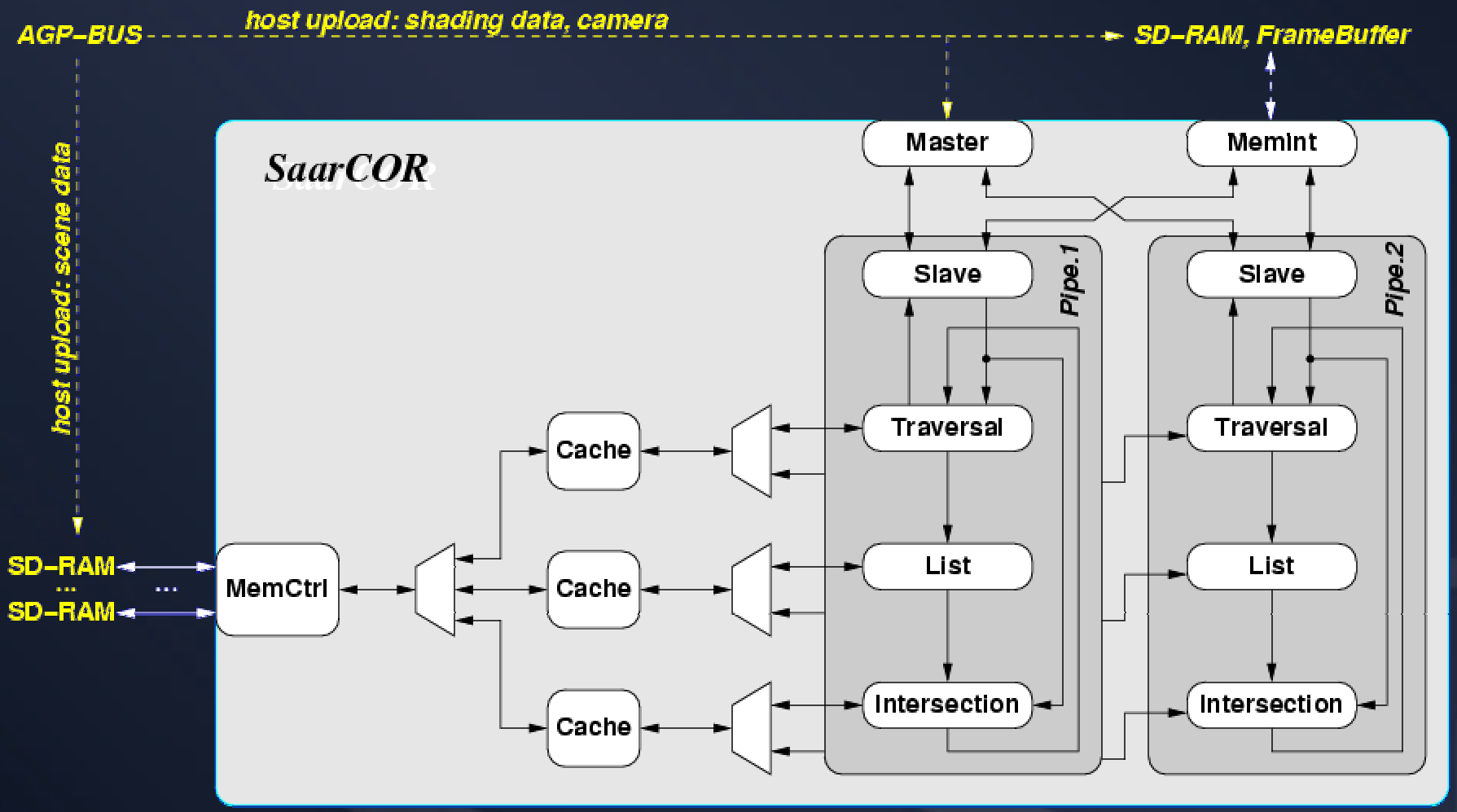
SaarCOR Architecture:
System overview



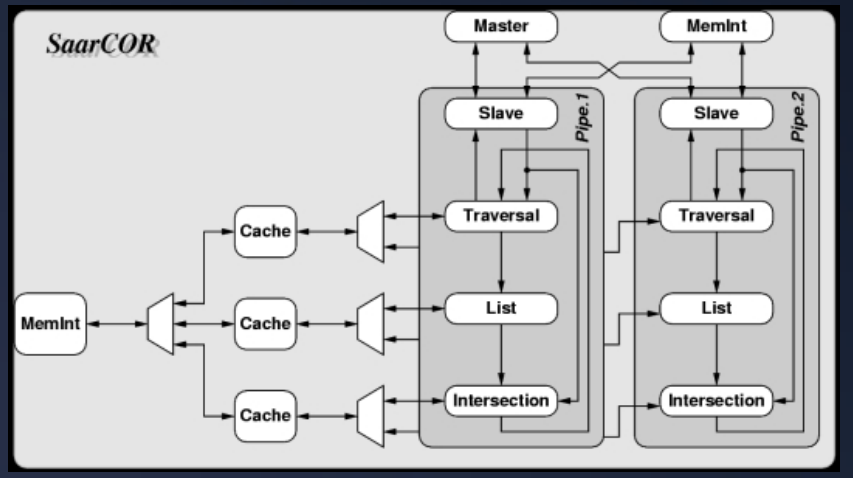
SaarCOR Architecture:
System overview



SaarCOR Architecture:
System overview



SaarCOR Architecture:
Features



- Scalable
- Fully pipelined
- Multi threading for latency hiding
- Simple communication pattern (no routing)
- Small independent functional units
- Asynchronous shading

Simulation

Simulation on register-transfer level

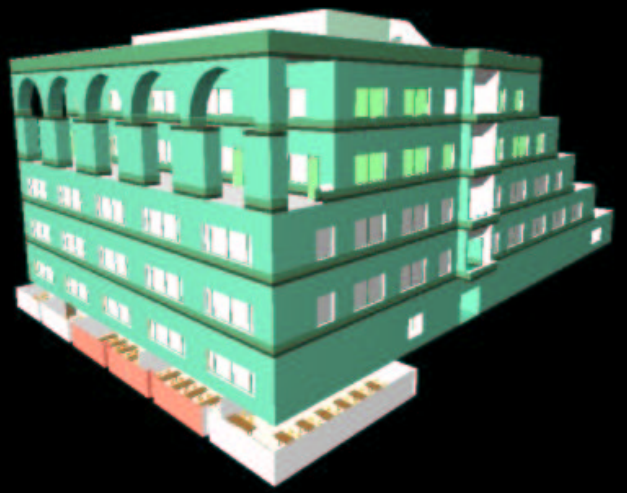
- Core @ 533MHz, Memory 64 Bit @ 133 MHz (simple SD-RAM, no DDR!)
- Each pipeline uses 36 simplified FP-units
- Standard SaarCOR:
  - 4 pipelines
  - 16 threads per pipe
  - 1 GB/s bandwidth to memory
  - 272 KB for caches
- Four pipes ~ ½ FP-performance of GeForce 3

Benchmarks:
Scenes

OpenGL-Like Shading:

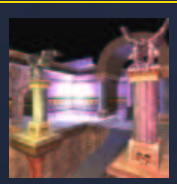
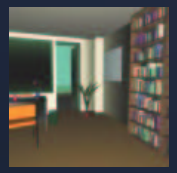
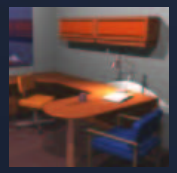
- No shadow rays
- No reflection rays

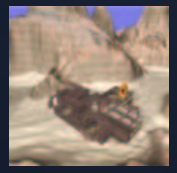
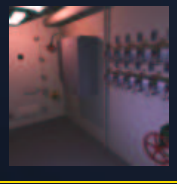
Full screen resolution
1024 x 768 pixel



Benchmarks:
Scenes (2)

Benchmarks:
Results

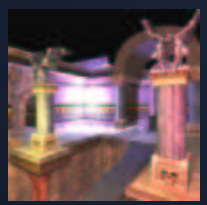
scene	frames per second 1 pipe	2 pipes	4 pipes	8 pipes	mrays/s 4 pipes
Quake3	27.20	54.45	111.12	225.39	87.4
Sodahall	28.88	56.71	113.22	225.67	89.0
Office	7.52	14.34	28.56	55.97	110.4
BQD-1	11.74	23.12	45.90	75.43	72.7
BQD-2	7.55	12.98	17.43	15.34	27.0
Cruiser	9.82	17.38	20.05	19.98	47.3

Quake3	35k tris, 768k rays	OpenGL-like shading
Sodahall	1.5m tris, 768k rays	OpenGL-like shading
Office	34k tris, 3.9m rays	three lights, reflections
BQD-1/2	2.1m tris, 1.6m rays	one light, reflections
Cruiser	3.6m tris, 2.4m rays	two lightsources

Today's CPUs: 0.5 – 0.8 mrays/s → factor of 100–200!

Benchmarks:
Results (2)



scene	1 pipe	2 pipes	4 pipes	8 pipes	
BQD-2	7.55	12.98	17.43	15.34	fps

working set	mem(nodes)	mem(tris)	#tris	% of total
BQD-2	663 KB	1.3 MB	37 558	1.8 %
Cruiser	315 KB	3.9 MB	112 359	3.1 %
SodaHall	230 KB	269 KB	7 651	0.5 %
Quake3	60 KB	122 KB	3 457	10.0 %

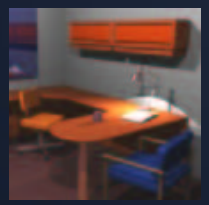
bandwidth	size of caches for nodes, lists, tris			
to memory	64,64,144	128,128,288	256,256,576	KB
1 GB/s	17.43	22.33	25.19	fps
2 GB/s	25.07	27.31	28.20	fps
4 GB/s	26.78	28.02	28.58	fps
hit-rate	67,87,85	78,88,92	81,88,94	%

All benchmarks: impact on performance by

Frame to frame coherence (reuse of cache)	< 1 %
Design of cache (direct mapped, 2–way, 4–way)	< 3 %

Benchmarks:
Results (3)






Efficiency of standard SaarCOR

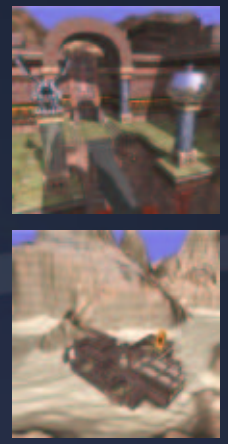
scene	1 pipe	2 pipes	4 pipes	8 pipes
Quake3	76%	76%	78%	79%
Sodahall	80%	79%	79%	78%
Office	71%	68%	68%	66%
BQD-1	73%	72%	71%	58%

16 threads → 32 threads: + 10%

Performance scales with number of pipelines, threads, cache size and bandwidth.

What about shading?

- Only visible triangles are shaded
  → no overhead due to overdraw
- Shading packets of rays exploits coherence
- BQD scene with bilinear textures
  - 14 MB for shading data per frame
  - 300 – 600 MB/s bandwidth
- Shading BW ~ Ray Tracing BW



Issues

On-chip memory of standard SaarCOR

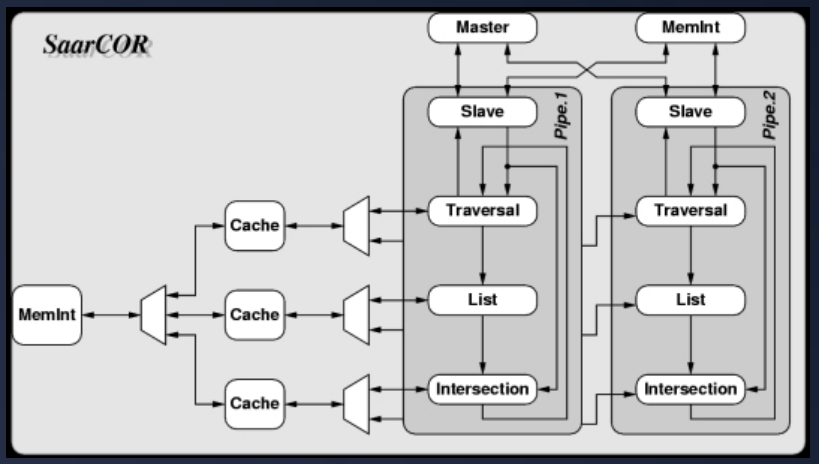
- Caches: 272 KB
- RF for rays: 288 KB
- RF for stack: 535 KB

Register level simulations only

Simple shading only

Conclusion

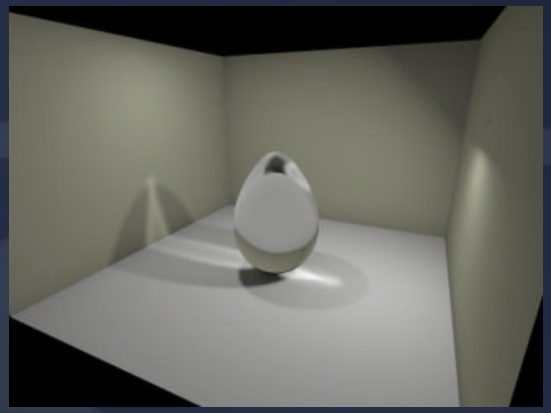
SaarCOR architecture



- scales well in the number of pipelines
- is highly efficient
- uses half the FP power of GeForce3
- requires very low bandwidth
- can hide almost all latencies
- provides full featured ray tracing
- frame rates of today's graphics cards

Future Work

- Complex shading
- Programmable shading
- API: OpenRT [Wald'02]
- Virtual Memory Management
- Large Models [Wald'01]
- Dynamic scenes [Wald'02]
- Global Illumination [Wald'02]
- Building a prototype



Thanks for your attention!

Questions?

graphics.cs.uni-sb.de
OpenRT.de
SaarCOR.de