Full Name: SAMPLE ANSWERS
Andrew ID:

15-719: Advanced Cloud Computing
Midterm Exam II
May 3rd, 2017

Total Time: 80 minutes. Manage your time wisely.

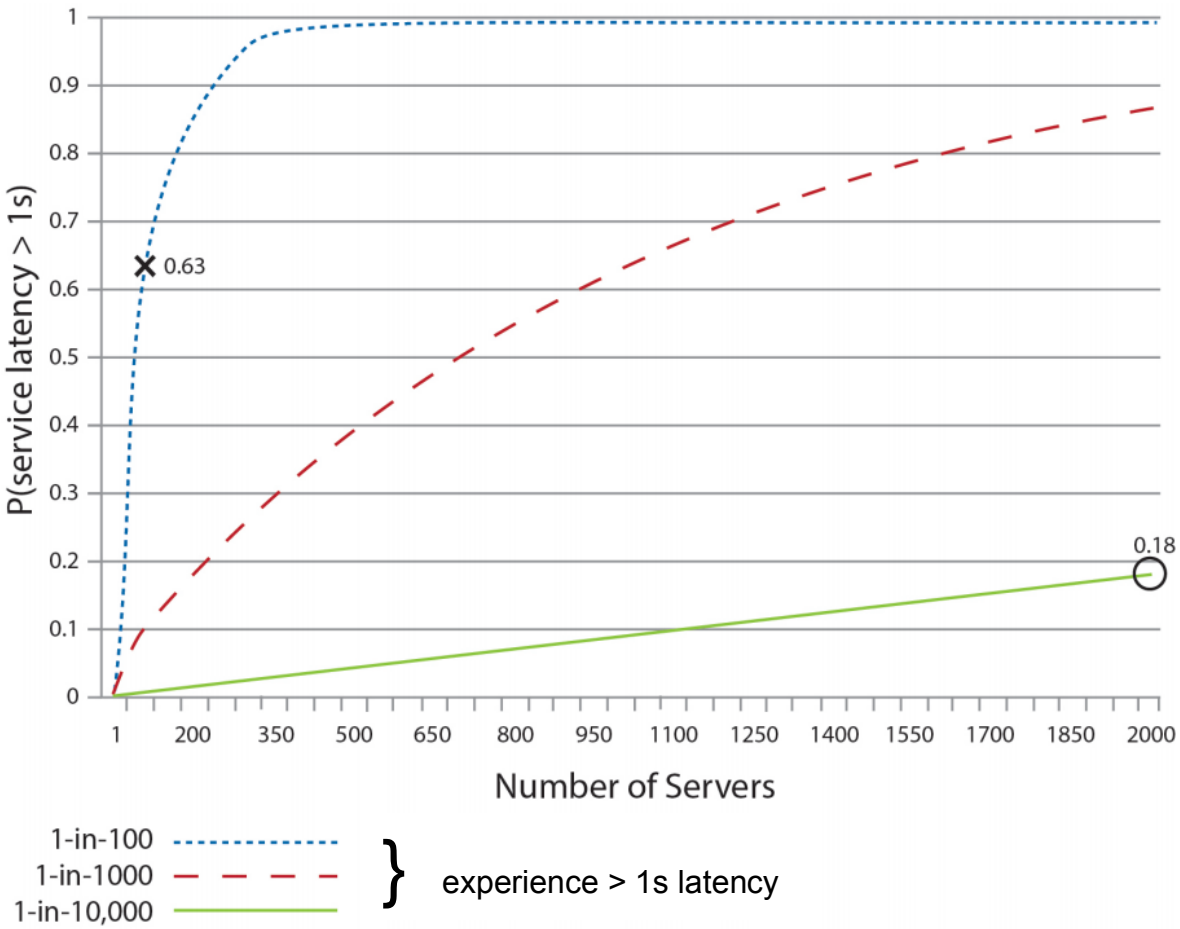
Instructions:

- Write your Andrew id on the top of every page in case they get separated.
- Make sure your exam is not missing any sheets. There should be **11 answer sheets**.
- Write your answers in the spaces provided below each problem. If you make a mess, clearly indicate your final answer.
- If you have to make an assumption, write it down clearly.
- This is a closed book exam. You are allowed one 8.5" x 11" sheet of notes.
- No electronic devices are allowed (**no** laptops/computers/smart-phones/watches).
- There are 70 points in 10 questions (+ 2 bonus points), but you should answer only **9.**
- **Question 10 is mandatory for all students that submitted project 3.**
- **Your final score is the sum of 9 answers; that is, out of 63 points.**
- **Cross out BELOW the question you do not want graded. You cannot cross out only a part of a question (we grade all parts of the questions you want graded).**

Question No	Max. Points	Earned Points
1	7	
2	7	
3	7	
4	7	
5	7	
6	7	
7	7	Most skipped
8	7	
9	7	
10	7	
Bonus	2 (bonus)	
Total	**63 (do 9 of 10)**	

1. **Warehouse Scale Computers: (Points: 7)**

   I. Explain the figure below, both “x” and “o” markers, and its implications on warehouse scale computers? **(Points: 3).**



A: Figure describes the impact on service level latency by tail latency.

- “x” in the figure: If a request must get responses from 100 servers in parallel, then 63% of the requests will take more than 1s.
- “o” in figure: if 1 in 10,000 requests experience over 1 second latencies at the single server level, then a service with 2,000 servers will see 18% of the requests taking over 1s.

Tail latency significantly impacts the latency of applications or services running on large scale warehouse computers. Techniques that mitigate tail latency, such as replication and redundant execution, become necessary to enable applications to run on large warehouse scale computers.

II. The following quote is from Barroso et. al., “A large application that requires many more servers than can fit on a single rack must deal effectively with these large

discrepancies in latency, bandwidth, and capacity."

Give an example of the source of these discrepancies in latency, bandwidth and capacity in warehouse scale computers and explain the implications on applications? **(Points: 4)**

A: Warehouse scale computers (WSC) are made up of clusters of racks, where each rack contains 10s of servers. Given the topology of a cluster, latency, bandwidth, and capacity differ significantly between servers, racks and clusters sometimes by orders of magnitude.

For example, an application could require many servers that span several racks in a large WSC. Hence, the application will have to read and write data from across the WSC. Accesses will either be local, rack-local or from within the WSC. Hence, the application will encounter large discrepancies in latency, bandwidth, and capacity. This increases the difficulty in programming applications for WSC. An application needs to account for, exploit and deal with large relative discrepancies on latency, bandwidth and capacity. For example, a MapReduce job attempts to locate map tasks nearby their input splits to take advantage of locality and reduce data transfer on the network.

2. Reliability & fault tolerance: (Points: 7)

I. A core concept in ACID transactions is Consistency. Another student in your class, Chaomin, says that data is defined to be consistent if and only if every copy of a replicated data set reports the same value when you read them all during a transaction. Explain why you do or do not agree with Chaomin. **(Points: 3)**

A: I would disagree because only a small percentage of the databases contain nothing but a replicated variable. Consistency in ACID means the database state conforms to some integrity rules defined by application. In the general case, the applications using a database have many more business rules for the data in a database. For example, the sum of all the bank accounts may be required to equal the sum of all the bank vaults plus all teller machines plus money loaned out to customers minus profit made by the bank. In general the consistency of a database are the business rules implemented by each and every transaction, and are not easily summarized. More simply, using ACID transactions, database software provides AID while the application code provides C.

II. Your internship company produces a pub/sub service (like LDAP or DNS). A co-worker, Nitin, says state machine replication can allow customers to read information with as little as one replica, so if you start with five replicas in different availability zones, the chance of all five being non-functional is so low that your company's publication service must certainly be highly available. Nitin further asserts that since quorum consensus requires a majority of replicas to be functioning in order to respond to any request, it is more likely to blocked by multiple failed machines and so it must be lower availability. Explain why you agree or do not agree. **(Points: 4)**

A: SMR requires all replicas to be functioning in order to make a change but only one replica to be functioning to read a value. Quorum Consensus requires a majority of replicas to be available to read or write a value.

The key assumption in the question is that a service is available if it is able to respond to read requests. This is not the most common definition of available, but in a pub/sub service like DNS, writes can take so long no one waits for them, but reads most be fast, so it is an appropriate definition of availability.

It is also the case that Nitin must be assuming fail-stop failures, or his first statement is also incorrect.

So assuming available is available for read, and fail-stop failures, then I agree with Nitin because quorum consensus blocks reads as soon as the number failures reaches half the number of replicas, while simple SMR can still respond to reads even if all but one replica has failed.

I would definitely remind Nitin that for writes, the opposite is true – one failed node will stop SMR write responses, while quorum consensus will be available for writes until half of the nodes are failed. So, if I define availability as available for read and write (that is, if I reject the premise of the question), then SMR is less available than Quorum Consensus.

It is also possible to reject the assumption of fail-stop failures. If you choose to assume Byzantine failures, then the reasoning Nitin gives is inappropriate. Moreover, in a Byzantine failure version of this question, simple SMR is not viable. One might declare a version of Quorum Consensus (that is, Byzantine fault tolerance) would be more available than an inappropriate simple SMR.

3. Tail latency & interference: (Points: 7)

I. Imagine a service in which every one of 10 servers must provide 1/10 of the answer to each client's request. Imagine that the operators of that service notice that each of the 10 servers is slow in providing answers for 1% of client requests, due to an intermittently occurring SSD cleaning process. If the key designer of that service proposes that the 90th percentile of client response times could be reduced by coordinating the cleaning process across the 10 servers, so it always occurs at the same time on all servers, would you agree or disagree with her? Explain your answer. **(Points: 3)**

A: I would agree. Approximately 10% of client requests are slowed by the SSD cleaning in the first scenario (a different 1% by each of the 10 servers), but only 1% of client requests would be slowed if the SSD cleaning occurred at the same time on all servers.

Some students correctly noted that, in the scenario described, only 1-(.99)$^{\wedge 10}$ ~= 1-.904 (or 9.56% of requests) are too slow because of SSD cleaning. That is, the 90th percentile is already fast. So, one could argue that the 90th percentile would not be

improved by having SSD cleaning at the same time on all servers. A counter-argument would be that there would remain very little "room for error" in terms of other slowdown effects, without such a change, but we accepted the argument if supported by the mathematical justification.

II. In some large-scale data processing systems, such as map-reduce, speculative task replication is used to mitigate tail latencies of task execution times. Identify one cause of slow task execution that this approach would address and one that it would not. Explain your answer. **(Points: 4)**

A: It would address issues such as a slow machine or slowdown due to an intermittent issue (e.g., the SSD cleaning example of part 1 above). It would not address issues that occur every time the task is executed, such as something related to the amount or content of data being processed by the task.

4. Geo-replication: (Points: 7)

I. Victor, the old man in your engineering team says that geographically triplicated data centers ensure that if one data center is partitioned from the others, then users connected to the data centers that can communicate will continue to receive service. Shelley, the young intern from CMU, replies that this is not the most important reason for geo-replication in the internet services world. Give at least one reason that Shelley might argue is more important than availability during partition and explain how she could defend this reason. **(Points: 3)**

A: The most important reason for geographical replication is to ensure that users everywhere have low-latency access to data, because the communication distance user-to-replica is usually lower than user-to-central-location.

II. Victor, in another conversation with Shelley, says that the CAP Theorem has been used to justify incorrect answers far too much, and that all services should provide linearizable order in order to make systems more likely to be correct. Wyatt Lloyd, the author of the causal consistency class reading, happens to be consulting and disagrees with Victor for two reasons. Give and explain at least two ways Wyatt disagrees with Victor. **(Points: 4)**

A: Restating and explaining the CAP Theorem was not an answer.

Wyatt would agree that in the abstract, the stronger the consistency model (linearizable is stronger than serializable which is stronger than causally consistent which is stronger than eventually consistent) the easier it is for programmers to form correct and safe assumptions about the data in the service.

But Wyatt would make at least two observations.

The first is that linearizable is stronger than serializable and that this additional strength is perhaps too expensive for its value. Specifically, linearizable ensures no covert channels -- transactions are not only equivalent to a serial ordering, but

that no external observer sees an ordering that is not equivalent. This requirement that external visibility be treated as communication forces real time clocking, which is expensive and restrictive – slowing down the system more than many think is worthwhile. Wyatt's second observation is that the only changes the system should block on are the ones that have a communication path to the transaction wanting to make progress. This is his causal consistency – block if there is any missing or stale information that might effect the computation, but don't block on changes that cannot impact the computation.

5. Security: (Points: 7)

I. Service level agreements (SLA) are at the core of virtualized computing. Give and explain at least one SLA property that you think the projects in this class have assumed EC2 provides but that you do not believe is guaranteed. **(Points: 3)**

A: The most basic SLA that all 719 project designs have assumed is that the instance name, m, r or c, and large or xlarge, hold meaning and promise different performance capabilities. AWS, like all cloud providers strives for this assumption to be correct, and might even discount your bill if you showed evidence of this not being true, but they make no guarantee.

Other examples include:
- performance isolation – the project assumed each instance would get the expected performance regardless of what else is happening in co-located VMs or common traffic paths,
- traffic pricing in availability zones – the project expected no charges from communication among nodes in a Spark job in one availability zone; the script given by course staff did not specify the SLA correctly, allowing AWS to locate an instance in another availability zone and charge extra for data transfer

II. Attestation is a mechanism used with Trusted Platform Modules (TPM). Describe the critical property attestation provides, the key assumption it has to make for attestation to work, and explain how it achieves this property. **(Points: 4)**

A: TPM promises that the code an external user expects to run in the TPM on the cloud node will either be the code that runs in the TPM or the external user will be easily able to detect that it is not. The key assumption is that the manufacturer of the TPM has designed hardware and a digital signature chain that the cloud provider cannot tamper with. SGX, Intel's TPM, achieves this by having the hardware hash the entire contents of the code and data space available to the TPM at the moment of creation (as the attestation hash), sign it with an Intel provided secret key and allow specific commands into the SGX to sign responses including this attestation hash. The remote user can know the code that should be running, Intel's public key for the TPM and include nonces in its challenge message to ensure that the only code running in SGX is the intended code.

6. Networking: (Points: 7)

I. In networking terms, layer 2 is the data link layer (e.g. Ethernet), layer 3 is the network layer (e.g. IP), and layer 4 is the transport layer (e.g. TCP). Many cloud implementations prefer to virtualize the network at layer 2. Give and explain one advantage of virtualizing the network at layer 2, and give and explain one disadvantage for virtualizing the network at layer 2. **(Points: 4)**

A: At layer two, LAN and VLAN bridges and switches will broadcast packets to reach the destination if needed, and dynamically build tables of where in the connected LAN/VLAN an addressed machine is attached to reduce the need to broadcast. So changes in virtualized LAN connection, or VLAN membership, are transparently supported, and efficient if the mapping tables are large enough. Unfortunately, the mapping tables have fixed size, so large virtual LANs may not be efficient and end up broadcasting too much traffic.

Other advantages of layer two include cost – layer two devices are less complicated and less expensive than higher layer devices; and security isolation, because VLAN cause traffic sent through real shared hardware by unrelated applications to be filtered so misbehaving unrelated applications can be denied the chance to listen in.

Layer three, on the other hand, does not use broadcast, but instead uses routing tables based on IP addresses. These routing tables allow traffic shaping to be done to deliver differential service to different flows and machines. This enhances the ability of the virtual network to do performance isolation or to over network performance service levels compared to layer two virtualization. However, layer three equipment is more expensive, and end-to-end connections often break when virtual machines are migrated between nodes whose underlying hardware has different IP addresses.

II. Tunneling, virtual private networks and virtual circuits are techniques for influencing the route a packet takes through a multi-path network. Give and explain one challenge or problem occurring in cloud computing that can be solved using one of these techniques. **(Points: 3)**

A: Customers of the cloud may want some machines in their non-cloud data center to be available to computations in the cloud. Using a virtual private network allows these non-cloud nodes to behave like they are located in the cloud datacenter.

Virtual circuits are used to performance isolate some traffic flows from other traffic flows so the cloud can offer a service level agreement that promises specific network bandwidth levels.Tunneling can be used to increase parallelism for traffic going from one part of the network to another, if the actual hardware is multiply connected. Instead of all data going through one path, it can be partitioned into many flows, each tunneling through a different midpoint router, so that the total traffic can exceed the bottleneck going through any specific router.

7. Verification (Points: 7)

I. Ironclad applications achieve end-to-end security through verification.

Give and explain the strongest property that Ironclad provides to a user deploying an application in the cloud and the assumptions Ironclad (and these users) is making about the remote servers in order to have these properties when executing in the cloud? **(Points: 4)**

A: The strongest property is end-to-end security for code run in remote servers, including some classes of service level agreements, based on verifying the actual machine code against the security specification. It allows the application writer to not have to trust the cloud provider, their infrastructure including hypervisors, the virtual machine OS, its libraries, and drivers. It does rely on the assumption that the hardware itself is trustworthy and correct, and uses some kind of TPM to attest to the machine code running on the hardware.

II. Give and explain the most important potential hindrance or challenge blocking wide adoption of such techniques? **(Points: 3)**

A:

- Development effort, difficult and costly, the ratio of verification code to actual code is still high.
- The potential performance degradation.

8. Diagnosis via Monitoring and Tracing (Points: 7)

I. Sampling is a common technique for reducing the costs associated with request flow tracing. Your colleague has proposed a design in which, at each trace point, a trace record is stored a configured percentage of the times that said trace point it is visited. What problem, if any, would arise in using a trace collected in this manner? **(Points: 3)**

A: It is likely the few full request flows will be captured by the scheme scribed. Indeed, a full request flow will only be captured in the very lucky case that, at every trace point visited by that request, the decision of whether or not to store a trace record is positive. (This is why most sampling systems make the decision once, at request arrival time, and carry the decision with the request in the form of a request ID that is NULL is no records should be stored.)

II. In many clusters, performance counters (e.g., of CPU time used, number of I/O requests, and number of packets sent) are recorded periodically (e.g., every 10 seconds). Give an example of a performance problem for which such counters would be insufficient to perform successful diagnosis. Explain your answer. **(Points: 4)**

A: Performance counters would not provide much help in diagnosing why any particular fraction of requests is performing poorly, because they aggregate information about the processing of ALL requests into the same measurement data. So, most examples of particular subsets of requests are reasonable answers, including requests from a particular user, requests about particular pages/files/items (whatever

the system serves), requests from a particular one of many client machines, etc. Another category of problems that performance counters may not be helpful for are those whose effects are mixed with non-problem behaviors, such as short bursts of slowness that are hidden within the counter sampling period.

9. Exploiting Spot Pricing (Points: 7)

I. Both the Proteus system you read about and elastic web services adapt the number of machines used over time. Explain how the primary consideration in the scaling policy (i.e., the metric or circumstance that causes the policy to change that number of machines) differs for these two use cases. **(Points: 3)**

A: For Proteus, the primary considerations relate to the time and cost for completing the collection of parallelizable work in the batch ML job. For an elastic web service, the primary consideration usually relates to using only the number of machines needed at each point in time to serve a time-varying client load within a specified latency SLO (e.g., 99% of requests within 200ms). So, the elastic web service can't use fewer machines than needed for the given client load, without failing the latency SLO, and can't use more than needed, because there isn't anything extra to do. Proteus could use extra machines to further parallelize work, which saves money if the extra machines are cheaper because the other more expensive machines are kept for correspondingly less time. [Extra explanation provided here for future studiers ;)]

II. Imagine that your company uses a distributed data processing system based on Hadoop, deployed on Amazon EC2 instances with all data stored in an HDFS file system running on those same instances. Your colleague argues that, because HDFS maintains three replicas of each block, it is safe to use spot instances instead of on-demand instances. Do you agree? Explain your answer. **(Points: 4)**

A: No. All of the spot instances could be taken back by Amazon at the same time, voiding the benefits of the replication. Replication in HDFS and other distributed storage assumes some degree of independence of failures, so as to provide enough time for repair before all machines holding replicas of a block fail.

10. Project 3 (Points: 7) MANDATORY IF YOU DID THE PROJECT

I. In part 2 of project 3, we used utility to measure the success of different scheduler policies. The utility of each job was calculated as U = 800 - job completion time (that is, execution stop time - job submit time). If a job takes longer than 800 seconds to be completed, it would earn zero utility. In addition, if a job took more than 1200 seconds to finish, it would have been killed by the system.

Since a job with a shorter completion time earns a higher utility, it would seem that minimizing job completion time (also known as response time) would be equivalent to maximizing utility. Do you agree? Why or why not? (**Points: 3**)

A: No.
For jobs that take longer than 800 seconds to finish, the utility of those jobs will remain 0 while the mean job completion time will keep growing according to the real job completion time
In addition, throttling max utility at 800 encourages a scheduler to give up jobs that produce no utility while using mean job completion time forces a scheduler to schedule all jobs

II. One way to understand and compare different scheduling policies is to draw a colorful figure showing the consequences of a series of scheduling decisions made for a certain job trace over the execution of the trace. We have created a few such figures shown below, based on project 3 part 1. These figures happen to be data taken from a real part 1 solution for FIFO-Random, FIFO-Heterogeneous, and SJF-Heterogeneous, but not necessarily in that order.

In these figures, the x-axis is the trace, each line "i-type-n" where i is the order of the job in the trace, type is MPI or GPU and n is the number of nodes needed. The lower portion of each figure shows the number of jobs in the queue at the moment each job is submitted. The blue bar shows the number of GPU jobs. The orange bar shows the number of MPI jobs. The upper portion of each figure shows what happens to that job, in seconds, where the top of the blue bar in the upper figure marks the time a job is submitted, the orange bar marks the time a job is pending in the queue, the green bar marks the run time for a job if it is scheduled to its preferred resources, and the red bar marks the run time for a job if it is scheduled to a non-preferred resources.

Identify the type of scheduler that produced each figure. Give a reason why you believe the scheduler you picked produced each figure. (**Points: 4**)

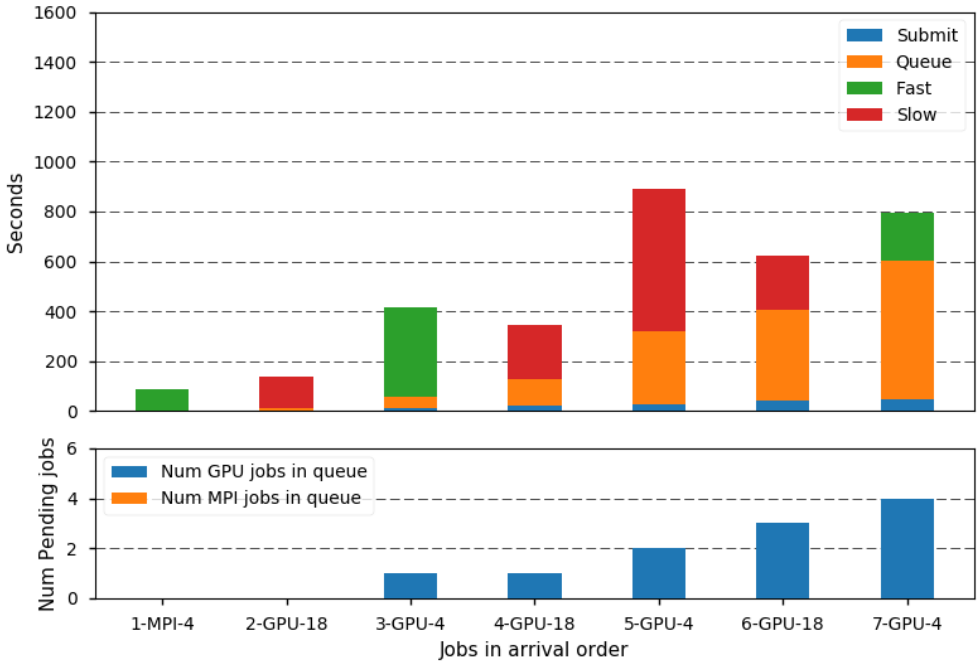


Figure 1

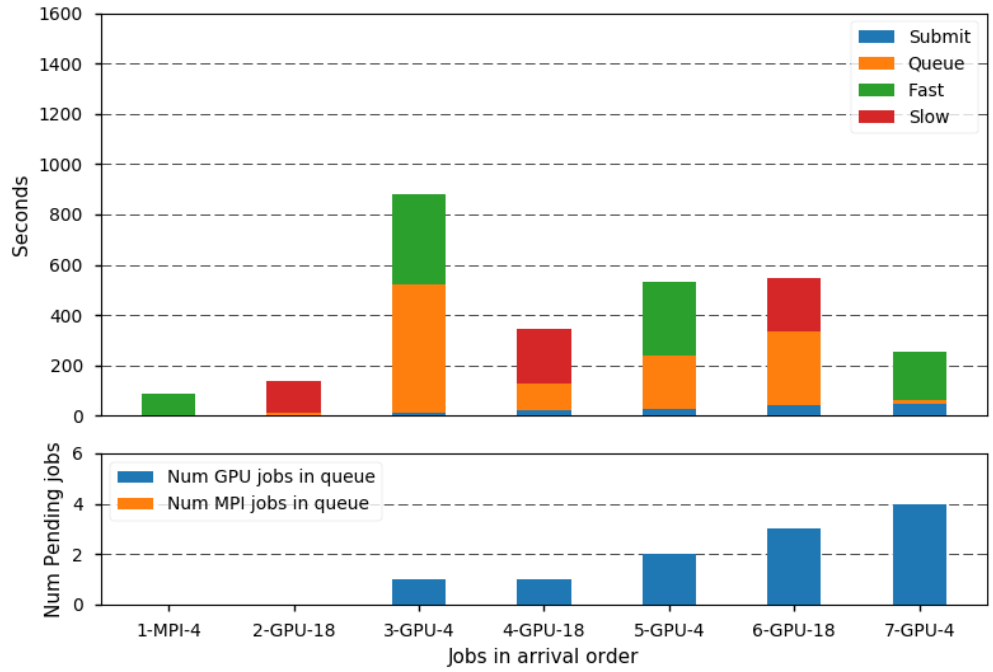


Figure 2

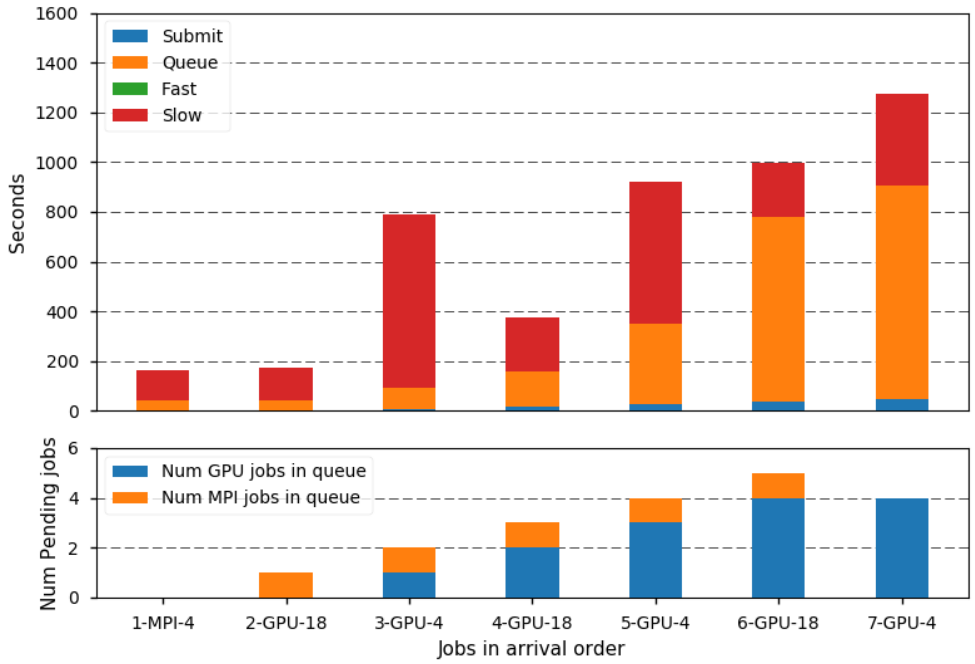


Figure 3

A: Figure 1: FIFO-heter since jobs are scheduled in their arrival order and to preferred resources where possible.

Figure 2: SJF-heter since jobs are scheduled to their preferred resources if possible but not in their arrival order. For example, job 3 arrived earlier but was scheduled after job 4, 5, 6, and 7.

Figure 3: FIFO-Random since jobs are scheduled in their arrival order but to random places, which is evident from the absence of achieving the preferred resources (green bars).