1. For: Distribution transparency can be useful because the user doesn’t need to know the details of how their data works(Endian, types…). The system maintainer can replace subsystems without rewriting everything on top. The user doesn’t need to know which server that data is running on or specify addresses. This means when it gets moved, no high-level programs need to be changed. Even if the data needs to move while it is used, that falls under the control of the system maintainer, not the end user. It is possible that data can be relocated for faster processing without the user needing to manually test for faster servers. The user doesn’t need to worry about concurrency, as the system should automatically handle any issues. Also if the system fails, it should be able to recover without the user bothering.

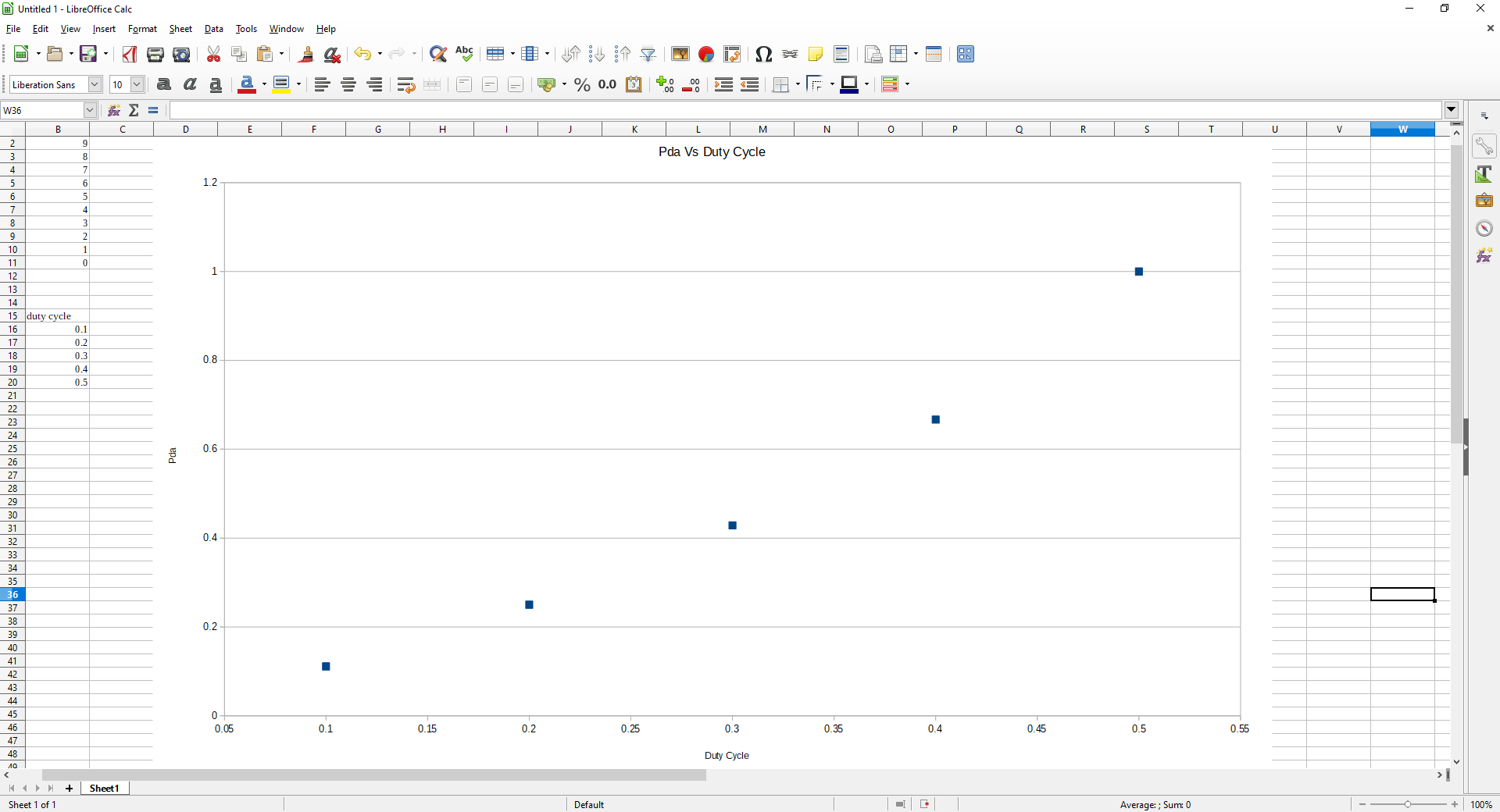
Against: The way the service processes the data may not be the same as the user intended, and it may not be evident where it differs. In the case of sensitive or private data, if you don’t know where the data is processed or held, you may not know if it was breached. It may also be accessible by people that were not intended initially if the organization has broad permissions. It may be difficult to diagnose failures, or it may have failed partially creating incorrect data, without creating any obvious errors.

1. For: There are many benefits of separating policy from mechanism. The first is in simplifying the interfaces for the majority of the users, instead of attempting to understand one large program and all of its detail, a user only needs to understand the interface used to work with it. This also allows the baseline functionality of the program to be improved without requiring each user to relearn the functionality.

Against: One drawback of separating policy from mechanism is the increased complexity in creation. Instead of creating a system where in internal mechanisms will always work in a predictable way, extra care has to be taken to ensure that updated or replaced components still function correctly. This also can increase the overhead as the program needs to manage a large number of subsystems and may not be able to take advantage of some optimization.

* 1. Service time can be decrease is various ways, depending on the source of the bottleneck. Raw computational power can be added by upgrading the machines that the program is running on. The program may be sped up if it was able to distribute parts of the program onto different machines. Some simple parts of the service may be able to be offloaded onto the users, such as data validation. Distributed systems may be able to be improved by reducing the number of messages sent between nodes, or by reducing the distance between nodes.

1. High performance distributed computing is the broad category of distributed systems focused on increasing computing power through various structures to divide work between a number of computers. Cluster computing is when a number of homogeneous computers on a high speed network are used. Grid computing is usually a cluster of clusters, where each cluster is generally independently managed, and may even be a grid of its own. Cloud computing is when the grid or cluster you are working on is managed by someone else entirely. Sensor networks and cluster computing both use a large number of devices to accomplish their tasks, however while cluster computing relies on a number homogeneous computers relatively close together, a sensor network can use a number of different devices at greater distances, and in more varied conditions. The computers used in a cluster are generally more powerful than sensors, and the network is generally more stable in a cluster than in a sensor network.
   1. The range of duty cycles that need to be considered is up to 50%active time, because after that both nodes will be active at the same time guaranteeing they can sync. However, the longer a node stays on, the more energy it uses. It can try only being active for a short time outside of it’s active cycle, however the longer it stays on the more reliable it can sync.



* 1. One way that synchronization can be improved for low duty cycles is to simply stay active when it is seeking another group, however this may mean wasting a large amount of energy. Instead after receiving a join message, it can wake out of cycle, during the active time of the other node, and join or send a join message then.

1. From an IT perspective service is providing functions that a separate entity can make use of without having to deal with the details of implementation. Cloud service is offering access to a services such as cluster computing, over the internet.
2. There are a few security vulnerabilities introduced by distributed computing. First the administrator needs to be more diligent if exposing their system to the world, putting a system on the internet can be a significant attack vector. There is also the need to decide what is supported, if you allow systems that are out of date you increase the chance a compromised machine gets access to your system. This becomes more problematic when working with another organization because that means not only are you exposing yourself, you also take on all the exposure of the system you are working with. If your company is very stringent with access to the system, but allows another company that is less strict, there may be unintended access. This also means that compatibility with the other system needs to be maintained, which could delay updates or risk locking out a significant portion of the users.
3. My laptop has an Intel Core i5-3210M. This processor has 2 cores, supporting a total of 4 threads. The maximum number of computational threads in MATLAB can be found using *maxNumCompThreads*, giving an answer of 2 because by default MATLAB only considers physical cores. Using *feature(‘numCores’)* gives a little more information.

|  |  |  |
| --- | --- | --- |
| Lead/Liaison | Chris Conway | cwc036@latech.edu |
|  | Yasmine Hines | yrh044@latech.edu |
|  | Doug Lindsay | dcl020@latech.edu |
|  | Carmen Nicholson | can024@latech.edu |
|  | Michael Towns | mst039@latech.edu |