CS 236r Final Project: Stable Compute Marketplace Problem with Multi-dimensional Preference Ordering

Gary Wu & Jessica Chen

Harvard John A. Paulsen School of Engineering and Applied Sciences

1 Abstract

There has been recent research in the intersection between computation and marketplaces, especially regarding machine learning. As machine learning models grow larger and larger, it becomes more and more difficult and costly for those without immense computational power or connections to train models, hence why some of the largest Large Language Models can only be trained by the likes of OpenAI and other leading AI/ML companies. Papers like DeepMarket and CrowdFL have proposed solutions to this with marketplace applications that allow machine learning compute suppliers and providers to lend out or rent compute of various forms (mobile phones, CPU's, GPU's, etc.)[1][2].

However, these papers have left out detailed implementation of the relevant pricing & matching algorithms that are crucial to these systems being implemented in practice. Given the complex nature of computational requirements for LLM training, we address the challenge for users to accurately report their preferences. Instead, users may report multi-dimensional preferences across simple factors like asking price and training duration, and our system explores the impact of stochastic reports or indifference groups on stability and social welfare. Our analytical and empirical results show that 1) stability rates exponentially decay as a function of stochastic preference reportings and preference group magnitudes; 2) stability rates are not a predictor of social welfare in our system; and 3) collapsing multi-dimensional preference inputs into one dimension for the Gale-Shapley matching algorithm can reliably generate revenue for our marketplace.

We have made our markets simulation code publicly available here: https://github.com/JessSanChen/computate-marketplace.

2 Introduction

Federated Learning (FL) and Distributed Deep Learning (DDL) are techniques that have gained massive popularity over the past few years, as they can enable the training of larger and more performant models given restrictions. FL can solve both compute and data scarcity problems while retaining privacy, and DDL largely enables better use of computational resources. In either scenario, the assumption is that one is able to recruit willing clients (in FL), or already owns all the compute they need (in DDL).

Parties that have a wealth of connections or the resources to access large amounts of compute have an inherent advantage in training the best ML models. We envision a future in which marketplaces can power the democratization of ML training, but this currently doesn't exist.

However, prototypes do, and there have been 2 papers published in the past few years that are imagining what these marketplaces may look like. Interestingly, these papers focus solely on the system and application design of such marketplaces, and do not research how topics like matching algorithms and dynamic pricing can fit into these systems.

Our team is particularly interested in the matching problem that exists in these systems. For one, it seems that such systems are highly **asymmetric** in terms of the supply and request sides, with many times more suppliers than requesters. Additionally, the criteria for which a compute supplier can participate in a requester's training run is generally quite strict (is having a GPU required or does a mobile phone suffice?). Further, factors like training duration (how long a supplier is willing to rent out their compute and how long a requester expects each training round to be) must also be taken into consideration.

A previous assumption in FL systems, for example, is that the utility that clients receive from participating is that they can receive a performant model. We introduce a new payment assumption to FL and outsourced DDL training, in the sense that the utility an agent might get from renting out their compute is no longer solely based on receiving the end model. In the future, we believe many participants in these systems won't find much utility in receiving a model, and may prefer direct payment instead.

2.1 Proposed Solution

All of these factors motivate research on a sufficiently tailored matching algorithm to address the challenges of this specific marketplace. We propose a marketplace that follows this structure:

2 sides: compute suppliers and compute requesters. Compute requesters want to train a model, and don't have the computational capacity to do so, so they are willing to pay suppliers that are willing to compute. Compute suppliers may have spare CPU's, GPU's, mobile phone's, etc. that they are willing to lend out to perform computation, in exchange for payment. For our work, we consider two heuristics that influence each agent's preferences: pricing and training duration.

On the compute requester side, pricing refers to how much they prefer to pay a supplier for a job, and training duration is how long they expect training rounds to be, given a device. For suppliers, pricing refers to expected payment received in accordance to the expected amount of time they are renting out their compute. Previous literature is sufficient in setting up such a system, however we now introduce the matching algorithm used to match these agents based on the two heuristics.

Specifically, we propose 2 multi-dimensional modifications of the Gale-Shapley algorithm to take into account how preferences may differ depending on the heuristic being measured (preferred pricing, training duration). Lastly, we design a simulation to analyze the stability of resulting matches and compare social welfare, stability, among different algorithms and indifference.

2.2 Goals

All of these factors motivate research on a sufficiently tailored matching algorithm to address the challenges of this specific marketplace. Our paper aims to achieve the following goals: 1. Design the multi-dimensional matching algorithm, 2. Prove properties of the algorithm, 3. Design a simulation to analyze stability, 4. Compare social welfare & stability based on type of indifference.

3 Related Work & Literature Review

CrowdFL introduces a marketplace platform for crowd-sourced federated learning (FL), which enables privacy-preserving collaborative model training on mobile or edge devices [1]. It manages client selection, model training, and reputation. The system lists tasks and recruits clients for efficient FL model training, addressing the challenge of straggling devices. Suggestions for improvement include turning it into a double-sided matching problem to enhance strategy-proofness in client and server reporting.

DeepMarket is an edge computing marketplace that uses a distributed TensorFlow execution capability [2]. Users lend computational resources and earn credits, which can be used to rent resources from others. Its goal is to utilize idle resources in the market, matching them with deep learning jobs. The system evaluates machine configurations and proposes improvements like allowing users to specify lending prices. However, it leaves out the implementation of any matching algorithm for future work.

Weights in Stable Marriage Problems introduces the notion of stability in stable marriage problems in which preferences are not strict but rather weighted, i.e. man m1 prefers woman w2 over woman w1 with weight x [3]. The paper focuses specifically on manipulation opportunities as a result of this weighting, however differs from our eventual model in that the weights are at the individual level, rather than at the heuristic level.

Two-Sided Matching with Indifferences addresses complexities in two-sided matching problems in which agents have indifferent preferences, a scenario often neglected in existing literature [4]. The paper challenges the assumption that agents have strict preferences, noting that indifferences are common in real-world settings, like job markets and school admissions.

4 Model

Consider a marketplace with: m compute requesters and n compute suppliers. We can represent them as X and Y respectively:

$$X = \{X_1, X_2, ..., X_m\}, Y = \{Y_1, Y_2, ..., Y_n\}$$

We introduce the notion of two heuristics for matching suppliers with requesters: **training duration** and **pricing**. In general, a good matching is one in which both parties are happy with the expected amount of time to train and also that there is agreement on fair payment.
Each agent is initialized with preferred a training duration and price for completing jobs, both of which are drawn from normal distributions.

We examine two ways of utilizing these heuristics to modify the original stable marriage problem into a mulit-dimensional one, and provide two multi-dimensional matching algorithms.

4.1 Weighted Preferences

More specifically, each agent spawns with a set of values:

$$X_i = \{D_{xi}, P_{xi}, w^1_{xi}, w^2_{xi}, S_{xi}\},\ Y_i = \{D_{yi}, P_{yi}, w^1_{yi}, w^2_{yi}, S_{yi}\}$$
$$0 \leq w^1_x, w^2_x, w^1_y, w^2_y \leq 1 \text{ and } w^1_x + w^2_x = 1,\ w^1_y + w^2_y = 1$$

D represents the agents preferred training duration, P represents their desired pay amount, w^1 is the amount of weight the agent places on training duration, and w^2 is the amount of weight the agent places on price in the weighted preference setting. We use S_i in the strict preference setting, which denotes the strict preference between training duration vs pricing. If an agent places a strong emphasis on training time (let's say a supplier wants low training times so they can be matched with many different requesters), then their matching should reflect this preference.

Algorithm 1: Pre-processing → Create Ordering from Weighted Preferences for X

Input: X, Y
Output: $R_X = [[], ..., []]$

R_X = [[],...,[]]
for *a in range(1,m+1)* **do**
    r1, r2 = [], []
    t1 = Y.sort(key = abs($X_a.duration - Y_j.duration$))
    t2 = Y.sort(key = abs($Y_j.price$))
    for *i in range(1,n+1)* **do**
        r1[i] = t1.index(Y_i)
        r2[i] = t2.index(Y_i)
    for *b in range(1,n+1)* **do**
        $R_X[a].append(w^1_{xa} * r1[b] + w^2_{xa} * r2[b])$
    /* Get the weighted sum by looking at the agent's rank and also the weighting importance */
for *b in range(1,m+1)* **do**
    sort $R_X[b]$ in ascending order
Return R_X

In order to transform our model to accommodate the Gale Shapley algorithm, we run the above algorithm to create an ordered preference from weighted preferences. We do the same for agents in Y, such that we now have two arrays R that contain each agents ordered preferences for each group.

This collapses our multi-dimensional preference system into a 1-dimensional one. Now with both preference orderings, we can simply run the Gale Shapley algorithm on both preference orderings: GS(R_X, R_Y) to receive a stable matching.

4.2 Multi-Dimensional Strict Preferences

Let's now consider the situation in which we create two strict preference orderings for each agent, one for each heuristic mentioned.

$$P_X = \begin{bmatrix} P^1_{X1} & P^2_{X1} & ..... & P^n_{X1} \\ P^1_{X2} & P^2_{X2} & ..... & P^n_{X2} \end{bmatrix}$$

$$P_Y = \begin{bmatrix} P^1_{Y1} & P^2_{Y1} & ..... & P^m_{Y1} \\ P^1_{Y2} & P^2_{Y2} & ..... & P^m_{Y2} \end{bmatrix}$$

Where the first row represents the strict preference ordering based on heuristic 1 and the same for the second row and heuristic 2, for example with agents X_1, Y_1:

$$P_{X1} = \begin{bmatrix} Y_2 & Y_4 & Y_1 & Y_3 \\ Y_4 & Y_1 & Y_2 & Y_3 \end{bmatrix}$$

$$P_{Y1} = \begin{bmatrix} X_2 & X_1 & X_3 \\ X_1 & X_2 & X_3 \end{bmatrix}$$

Our pre-processing algorithm is the following:

Algorithm 2: Pre-processing → Create Ordering from 2D Strict Preferences for X

Input: X, Y
Output: R_X= [[], ..., []]

R_X = [[ ],...,[ ]]
for *a in range(1,m+1)* **do**
    r = [ ]
    if $S_{Xa} == TrainingDuration$ **then**
        $r = Y.sort(key = abs(Y_j.price))$
        $r = r.sort(key = abs(X_a.duration - Y_j.duration))$
    `/* First sort by the less important heuristic, and then sort based on the more important heuristic */`
    else
        $r = Y.sort(key = abs(X_a.duration - Y_j.duration))$
        $r = r.sort(key = abs(Y_j.price))$
    $R[a].append(r)$
Return R_X

We do this for agents in Y as well, and then plug R_X and R_Y into Gale Shapley, returning a stable matching.

5 Stability

To show that our matching algorithm for weighted preferences is stable, we need to show that our pre-processing step (turning weighted preference into ordered preference) outputs an agent's true preferences. From there, we nicely inherit the standard Gale-Shapley algorithm's stability, with respect to the input preferences.

There are 3 cases to consider for each agent, let's use agents in X for example:

1. w_x^1 or $w_x^2 = 1$: in this case, we are maximally weighting the preferred heuristic. Suppose $w_x^1 = 1, w_x^2 = 0$. Then the weighted sum collapses:

$$w_x^1 * r1(Y_n) + w_x^2 * r2(Y_n) = r1(Y_n)$$

and the sorted list of weighted sums is just the ranking of members in Y based only on their training duration. The returned preference ordering thus matches the agent's true preferences based on our pre-processing algorithm.

2. $w^1 > w^2$ and $w^1 < w^2$: in this case, let's consider 2 scenarios: one where the ranking for a particular agent is the same across both heuristics and one in which the rankings differ.

1. When an agent has the same ranking across both heuristics, then the weighted sum becomes

$$w^1 * a + w^2 * a = a * (w^1 + w^2) = a$$

which then correctly just compares the rankings as is across agents.

2. When the agent rankings across the heuristics are different, then the weighted sum reflects some scalar that we can then use to compare against other agents, and take as true preferences.

3. $w^1 = w^2 = 0.5$: in this case, any preference between either of the heuristics will be the same as taking into account the rankings at face value without adding the 0.5 weight, and thus will reflect the agent's true preferences. For example, if a particular agent is ranked 2 for heuristic 1 but 3 for heuristic 2, this is just calculating the average of the rankings, which correctly denotes the agent's comprehensive preference.

In all cases, we can take the weighted preferences along with rankings based on both heuristics, into some true preference that we can then run Gale Shapley on. This then returns a stable matching with respect to the true preferences.

6 Simulation

The main purpose of our simulation is to reach empirical results on the design decisions behind our proposed computational marketplace that are otherwise difficult to prove analytically in our current scope.

We no longer assume that our agents accurately know their preferences.

Since the computational requirements of training LLMs are sometimes complex to predict, agents may have imperfect understanding of their preferences. On the model requester side, they may have an idea for the model accuracy they are seeking or an approximate range for the expected time to convergence, but this is difficult to intuitively quantify and costly to know from experimentation. Furthermore, the model requester may not have fully accurate understandings of the market price for the demanded computational resources; it is not feasible for agents to know what prices third-party cloud providers are offering for the same computational resources, so requesters may post a certain asking price while still be willing to pay more should the demands call for it.

On the other hand, the training clients may not accurately know the availability of their resources at the time of matching. Since it is on the training client to post their availabilities and await proposals from model requesters, it is possible that by the time of training, their availability have changed by a small amount. Furthermore, the training clients' posted asking prices might not fully encapsulate prices they would be satisfied with; again, as a way to take into account potential market fluctuations but without adding the complexity of equilibrium pricing into our project, we have added stochasticity to pricing preference as well.

We no longer assume that our agents have strict preferences between various product bundles.

Instead, there may be groups of bundles within which an agent is indifferent between the bundles. These are called *preference groups*, which are discrete groups of one bundle or of many. Building

on our previous point that the computational requirements of LLMs are difficult to accurately predict beforehand, some agents may find it difficult to distinguish between minute differences of various offerings. In our simulation, we create groups based on certain thresholds of ϵ. A normal ceiling function would have $\epsilon = 1$; here, we vary ϵ to see how the magnitude of these preference groups may affect the stability rate of the overall market.

This preference grouping may also be extended to other forms of indifference. For example, if a model requester is rather confident in their computational needs and simply want resources for a duration t, then given that they are willing to pay price p for duration t, they may be just as willing (but no more) to pay price p for duration $2t$. This indifference creates groups that are entire regions of inequalities. For model requesters, any bundle with a greater duration may yield the same utility. For training clients, the reverse is true. While we do not simulate this in our paper, we hope our results may provide insight into how greater areas of indifference would perform.

We still assume we know how agents may calculate one-dimensional preferences from multi-dimensional preferences.

In this paper, we only consider two dimensions of preferences: asking price and duration of training. These two dimensions are potential tradeoffs of each other: a model requester may be more willing to pay more if they are guaranteed a longer duration for training; conversely, a training client may free up greater availability of resources if that meant receiving greater payment. Therefore, an agent may get equal utility for drastically different bundles. Like in many economies, we have created indifference curves to express these tradeoffs that an agent may face.

In our analytical section, we focused on one simple scoring function. A *scoring functions* yields a one-dimensional from multi-dimensional preferences. Here, we want to show that more complex forms of calculation may yield similar results. We do not do extensive exploring, nor do we find bounds for what "more complex" may entail. Still, we wanted to consider more realistic utility functions to better demonstrate these tradeoffs and indifferences. Hence, our simulations look at both weighted-ranking and utility scoring funcitons.

6.1 Setup

Some points to highlight in our simulation include:

1. A `Marketplace()` object is instantiated given the desired number of agents, some Gaussian generation statistics (mean and standard deviation of duration and price), relevant algorithms (calculating reported preference orders, underlying true preference orders, and

2. For any datapoint, we run a default of 100 rounds before averaging. We solve for the static problem, which involved matching all agents currently in the market.

3. Price is calculated as a linear function of duration, wrapped in a Gaussian distribution. There are better ways to calculate pricing (see equilibrium pricing by Professor Parkes and others), but we have limited our scope to a fixed willingness-to-pay.

4. Our implementation of the Gale-Shapley algorithm is requester-proposing. Final matching price is based on the lender's asking price. Together, both requesters and lenders have an

advantage (Pareto-optimal for the requesters and price-matching for the lenders). Market revenue is then the sum of the lenders' asking prices. Aggregate spread is the sum of all the differences between lender prices and renter prices. In the future, we may explore more effective pricing mechanisms.

5. Price preferences are sorted ascending or descending by value, while duration preferences are sorted by their distance from the given agent's asking duration. This assumption is made to encapsulate two ways other factors of resource requests may be parsed into preference orders.

6. The *weighted ranking scoring function* sorts the potential matches by either criteria, then take the weighted average of their rankings. The *utility scoring function* first normalizes the prices and durations, then take the weighted difference between the two factors (for renters, the price term is negative; for lenders, the duration term is negative).

6.2 Results

For this paper, we are interested in the stability rate and social welfare of our system. Stable systems do not contain blocking pairs, and have historically been shown to be a characteristic of more sustainable systems. We measure social welfare by market revenue and the pricing and duration spreads.

6.2.1 Stability Rate of Stochastic Preferences

In Figure 1, we find that the stability rate is an approximately exponentially decaying function of greater stochasticity for either scoring function. In other words, if we were to define stochastic preferences as a sample of the Gaussian distribution with its mean about the reported preference, then as the standard deviation increases, stability decays exponentially. We see similar results for both scoring functions.

(a) Weighted ranking scoring function

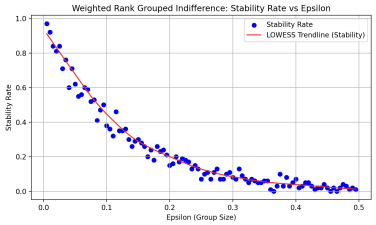
(b) Utility scoring function

Figure 1: Stability rate of stochastic preferences vs. report std

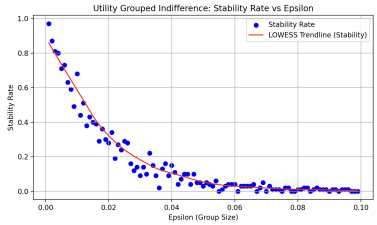
6.2.2 Stability Rate of Grouped Preferences

In Figure 2, we find that the stability rate is an approximately exponentially decaying function of greater grouping magnitudes ϵ. We see similar results for both scoring functions. This suggests again that stability of the Gale-Shapley algorithm follows a certain probabilistic trend when met

with a naive process for indifferent preference groups. Further work, including implementation of literature in this field, can be implemented into our computational marketplace.



(a) Weighted ranking scoring function

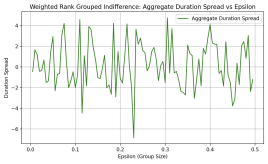


(b) Utility scoring function

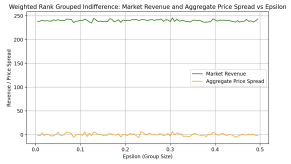
Figure 2: Stability rate of grouped preferences

6.2.3 Market Revenue and Pricing/Duration Spreads vs. Stability

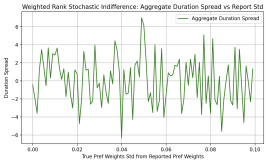
In Figure 3, we consider both group size ϵ and the standard deviation of report stochasticity as measures of stability. Then, we explore how measures of social welfare, such as market revenue and pricing/duration spreads, vary with lower rates of stability. As expected, we did not find clear trendlines, nor do we see patterns in greater variation. Stable matching, while potentially beneficial for long-term revenue stability and growth, does not optimize for revenue.



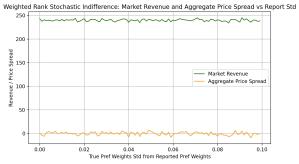
(a) Ranking Grouped-Pref Duration Spread



(b) Ranking Grouped-Pref Revenue/Price Spread



(c) Utility Grouped-Pref Duration Spread

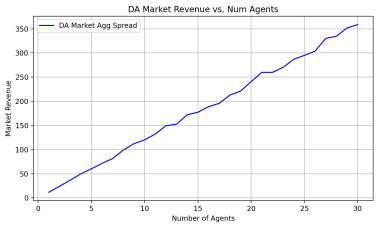


(d) Utility Grouped-Pref Revenue/Price Spread

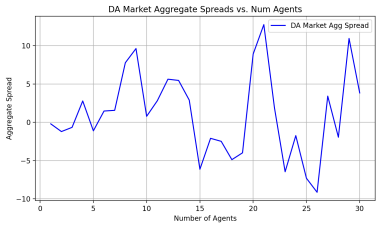
Figure 3: Social welfare as a function of stability.

6.2.4 Gale-Shapley Performance with Social Welfare

In Figure 4, we find that our proposed Gale-Shapley method with lender-based payment performs with room for improvement. While system revenue increases with respect to the number of participating agents as expected, we believe further work can be done to address the high variability in pricing spreads across agents. With such unpredictable price spreads, it is difficult for participating agents to lend or rent with certainty of their revenue or cost. Therefore, further work should be done to explore alternative pricing methods that may still provide higher rates of stability while also providing guarantees of pricing spreads.



(a) Market Revenue vs. Number of Agents



(b) Aggregate Price Spread vs. Number of Agents

Figure 4: Gale-Shapley social welfare performance

7 Next Steps and Discussion

This paper provides the framework for a two-sided matching mechanism for a marketplace for computational resources between lending training clients and renting model requesters. Our proposal shows that we are able to take accessible user preference inputs along multiple dimensions and output a matching, with certain empirical bounds for stability guarantees.

Through our simulations, we find that stability rate exponentially decays with increased stochasticity and increased magnitude of indifference grouping. Interestingly, we do not find correlations between stability rate and social welfare measures, such as market revenue and price/duration spreads.

Given the limited scope of our work, we propose a number of next steps that we would like to explore:

1. **Alternative pricing mechanisms.** There is no direct relationship between stability and revenue, and the high unpredictability of spreads for increased numbers of participating agents suggest the need for a more robust pricing system. If we were to continue assuming fixed pricing based on willingness to pay, we would like to explore pricing mechanisms specifically for matching problems. Further work could also explore equilibrium pricing methods to better simulate a market of supply and demand.

2. **Incorporating higher dimensions of preference attributes.** For this paper, we considered just two dimensions: price and duration of training. Further work should be able to extend our findings to higher dimensions when calculating scores. In particular, we are interested in latency factors (such as distance) and reputation scores (which rewards successful training and punishes dropouts or performance variability while training).

3. **Removing the assumption that we know how to calculate the underlying true preference order.** While attempts were made, we could not make progress in this aspect. In practical applications, it is very difficult to know the process an agent may take to convert multi-dimensional preferences into one dimension. While we looked at two different scoring functions, we believe this is a far more complex problem that will require greater statistic analysis.

Given the rise in demand of distributed training, particularly in the face of expensive third-party cloud providers, we hope to further contribute to this effort to create a sustainable marketplace for computation.

8 Acknowledgements

We would like to thank Safwan Hossain, our Teaching Fellow, for his flexibility in office hours and constructive feedback on our experiment design. We would also like to thank our classmates, for their willingness to answer silly Perusall questions and their dedication to each section's presentations. Finally, we would also like to thank Prof. Yiling Chen, for a semester of teaching and patience. We were both very surprised, and very honored, to have been allowed to take this seminar, and we are truly grateful for the opportunity to be exposed to such exciting topics and surrounded with such incredible minds. In particular, we were very appreciative for the guidance she provided as we defined our scope, explored literature, and overcame roadblocks.

Separately, we would also like to mention that we were so taken by incentive alignment problems that we also incorporated into another one of our final projects! In CS 243 with Prof. Minlan Yu, Gary and I incorporated incentive alignment in the federated learning client selection problem. Although we know there is still much to learn in this field, we hope to continue honing our skills and truly make a contribution to these topics on the frontier of research.

9 References

1. **Feng, D., et al. (2022)**, "CrowdFL: A Marketplace for Crowdsourced Federated Learning", Proceedings of the AAAI Conference on Artificial Intelligence, 36(11), 13164-13166. https://doi.org/10.1609/aaai.v36i11.21715

2. **S. Yerabolu et al. (2019)**, "DeepMarket: An Edge Computing Marketplace with Distributed TensorFlow Execution Capability," IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Paris, France, pp. 32-37, doi: 10.1109/INFCOMW.2019.8845247.

3. **Aytek Erdil, Haluk Ergin (2017)**, "Two-sided matching with indifferences", Journal of Economic Theory, Volume 171, ISSN 0022-0531, https://doi.org/10.1016/j.jet.2017.07.002.

4. **Maria Silvia Pini et al. (2011)**, "Weights in stable marriage problems increase manipulation opportunities", Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/2000378.2000402