

## 1 A Proof of $\overline{T_{exp}}(T; \rho)$ 's Trend with Varying $\rho$

2 In Section 3.2, Fig. 1(c) demonstrates that: Given input token count  $t = T > 1$ , the number of tokens  
 3 each expert processes on average  $\overline{T_{exp}}(T; \rho) = \frac{\rho^T}{1-(1-\rho)^T}$  decreases as  $\rho$  decreases. We prove this by  
 4 showing  $\frac{d(\overline{T_{exp}}(T; \rho))}{d\rho} > 0$  when  $T > 1$ .

$$\frac{d(\overline{T_{exp}}(T; \rho))}{d\rho} = \frac{d(\frac{\rho^T}{1-(1-\rho)^T})}{d\rho} = \frac{T(-\rho T(1-\rho)^{T-1} - (1-\rho)^T + 1)}{(1-(1-\rho)^T)^2} \quad (1)$$

5 Since  $\rho$  represents MoE sparsity  $\in (0, 1)$ , the original proposition is equivalent to proving:

$$\mathbf{F}(\rho; T) = (1-\rho)^{T-1}(\rho T + 1 - \rho) < 1 \quad (2)$$

6 Note that  $\mathbf{F}(\rho; T) \rightarrow 1$  as  $\rho \rightarrow 0$ . Therefore, if we can prove that  $\mathbf{F}(\rho; T)$  decreases as  $\rho$  increases  
 7 from 0 to 1, then the original proposition is proven. We demonstrate this by computing  $\frac{d(\mathbf{F}(\rho; T))}{d\rho}$ :

$$\frac{d(\mathbf{F}(\rho; T))}{d\rho} = \frac{d((1-\rho)^{T-1}(\rho T + 1 - \rho))}{d\rho} = -\rho T(T-1)(1-\rho)^{T-2} \quad (3)$$

8 When  $T > 1$ ,  $\frac{d(\mathbf{F}(\rho; T))}{d\rho} < 0$ . This confirms that  $\mathbf{F}(\rho; T)$  decreases as  $\rho$  increases, which proves our  
 9 original proposition: when  $T > 1$ ,  $\overline{T_{exp}}(T; \rho)$  decreases as  $\rho$  decreases.

## 10 B The Impact of Measurement Selection for Parameter Fitting on Modeling

11 As mentioned in Section 3.3, since GPU execution is dynamic in practice, and not all operators are  
 12 optimized to their theoretical limits, we introduced several parameters for relaxation. The values  
 13 of these parameters are then automatically determined by fitting to GPU measurements under least  
 14 squares criterion. The results shown in Figure 3 in Section 4.2 were obtained by fitting to 21 GPU  
 15 measurements. In this Section, we further explain:

- 16 • Fitting Details of the modeling shown in Figure 3. (Section B.1)
- 17 • How the modeling is affected by alternative measurement selection. (Section B.2)

18 The value of our modeling is twofold. On one hand, it achieves alignment with real measurements  
 19 with only a small number of simple parameters, thus validating the correctness of our theoretical  
 20 analyses. On the other hand, it provides the decomposition of various factors in the end-to-end results,  
 21 making the entire SD acceleration process explainable and transparent.

### 22 B.1 Fitting Details of the Modeling shown in Figure 3

23 We first explain how we select the 21 measurements. Due to GPU resource and time constraints, we  
 24 obtained a total of 228 GPU measurements across varying experimental settings, including 6 different  
 25 numbers of activated experts per token ( $K$ ), 2 draft lengths ( $\gamma$ ), and 19 batch sizes ( $B$ ). These  
 26 measurements are sorted first by  $K$ , then by  $\gamma$  within each  $K$  group, and finally by  $B$  within each  
 27  $(K, \gamma)$  combination, forming the total dataframe (df). We then uniformly sampled measurements  
 28 from this sorted dataset with a fixed stride, namely  $M = \text{df}[\text{begin}:\text{end}:11]$ . This sampling strategy  
 29 enables our selected measurements to contain different settings, making the modeling more robust.

30 The SD speedup function (namely, *ComputeSpeedup* defined in line 3 of Algorithm 1) is nonlinear.  
 31 To optimize its MSE, we employed the `scipy.optimize.least_squares` function with the Trust  
 32 Region Reflective (TRR) algorithm. TRR is an optimization method for bound-constrained nonlinear  
 33 least squares problems that combines trust region methods with reflection techniques. It constructs  
 34 quadratic models within trust regions and uses reflection strategies near boundaries to maintain  
 35 feasibility while ensuring convergence. The fitting process for these 21 data points is efficient,  
 36 completing in approximately 0.114 seconds. Our modeling incorporates 10 parameters requiring  
 37 relaxation, with their search boundaries specified as follows:

- 38 • *bias*: It represents the time required to load the dense parameters of the target model. We de-  
 39 note the model's non-FFN parameter count as  $V_{dense}$ . Consequently, the theoretical minimum

loading time can be calculated as  $bias_{min} = \frac{V_{dense} \times bitwidth}{peak\ memory\ bandwidth}$ . For the upper bound of the relaxation range, we set  $bias_{max} = 5 \times bias_{min}$ .

- $k1$ : It adjusts the intensity of the roofline effect of dense components. It should be larger than 0 to ensure the execution time increases as the token count increases. We don't set a definite upper limit for  $k1$ , as its value is affected by other parameters. Given the hardware with fixed arithmetic units, the execution time grows linearly with the token count in the compute-bound regime. As shown in line 6 of Algorithm 1,  $k1$  appears as a coefficient in the term  $k1 \cdot G(t; \lambda, s)$ , whose gradient in the compute-bound regime is  $k1 \cdot \ln(s) \cdot s^{\lambda RP}$ . As  $s$  approaches 1,  $k1$  needs to continuously increase to counterbalance  $\ln(s)$  that approaches 0.
- $k2$ : It represents the time required to load one expert. Given a target model, we denote the parameter count per expert as  $V_{exp}$ . Consequently, the theoretical minimum loading time can be calculated as  $k2_{min} = \frac{V_{exp} \times bitwidth}{peak\ memory\ bandwidth}$ . For the upper bound of the relaxation range, we set  $k2_{max} = 5 \times k2_{min}$ .
- $k3$ : It adjusts the intensity of the roofline effect of sparse components. Similar to  $k1$ , we set  $k3_{min} = 0$  and  $k3_{max} = inf$ .
- $draft\_bias$ : It represents the time required to load the dense draft model. We denote the draft model's parameter count as  $V_{draft}$ . Consequently, the theoretical minimum loading time can be calculated as  $draft\_bias_{min} = \frac{V_{draft} \times bitwidth}{peak\ memory\ bandwidth}$ . For the upper bound of the relaxation range, we set  $draft\_bias_{max} = 5 \times draft\_bias_{min}$ .
- $draft\_k$ : It adjusts the intensity of the roofline effect of the dense draft model. Similar to  $k1$ , we set  $draft\_k_{min} = 0$  and  $draft\_k_{max} = inf$ .
- $reject\_bias$ : It represents the fixed overhead when performing rejection sampling. Vllm reports its elapsed time during SD, and we denote the maximum across measurements as  $T_{rej}$ . We then set  $reject\_bias_{min} = 0$  and  $reject\_bias_{max} = T_{rej}$ .
- $reject\_k$ : It represents the incremental processing time in rejection sampling as the input token count increases. For simplicity, we set  $reject\_k_{min} = 0$  and  $reject\_k_{max} = T_{rej}$  just like  $reject\_bias$ .
- $\lambda$ : It represents the ratio of the empirical ridge point to the theoretical ridge point. Since memory bandwidth is typically less utilized than arithmetic units, we set  $\lambda_{min} = 0.2$  and  $\lambda_{max} = 1$ .
- $s$ : It adjusts the growing rate of execution time as input token count increases. Since  $s$  serves as the base of  $G(t)$ , it must exceed 1 to ensure monotonic growth. However,  $s$  should not be too large, as it would result in an excessively steep growth rate. In experiments, we set  $s_{min} = 1$  and  $s_{max} = 2$ .

## B.2 Exploration of Alternative Measurement Selection

In this section, we demonstrate the impact of varying the number ( $m$ ) of measurements used for fitting on the modeling results. Given that our model incorporates 10 parameters, a minimum of 10 profiling data points ( $m \geq 10$ ) are required to determine all parameters. We present the modeling fitting with  $m$  ranging from 10 to 228. The data selection method follows the stride-based approach described in the previous section, specifically  $M = df[begin:end:stride]$ . Measurement count  $m$  and  $stride$  satisfy the following relation:  $m = \lceil 228/stride \rceil$ .

We present the MSE values of different  $ms$  and their corresponding fitting figures in Table 1. We also list the distinct batch sizes involved in the selected measurements, which helps explain why some configurations show inferior model fit. Due to integer division constraints,  $ms$  are not continuous at larger magnitudes. Generally, the modeling fits well with the real measurements, except for  $m = 10, 12, 13$ . The reasons are as follows. When  $m = 10$ , the number of measurements equals the parameter count, resulting in insufficient data for robust fitting. When  $m = 12$  and  $m = 13$ , the distribution of the measurement data is biased. With stride-based selection, measurements at  $m = 12$  and  $m = 13$  demonstrate notable gaps in batch size coverage (specifically,  $m = 12$  does not include batch sizes greater than 40, while  $m = 13$  does not include batch sizes within 1~24). Their MSE values are larger than that of  $m = 11$ , despite the latter containing fewer data points for fitting. Based on this analysis, we recommend prioritizing uniform data distribution when selecting measurements, as this approach enables the development of more reliable models even with smaller datasets.

Table 1

$m$	$stride$	MSE	Figure	Batch Size Involved
10	25	2.216	1	[1, 12, 16, 20, 36, 40, 44, 60, 80, 100]
11	22	1.764	2	[1, 4, 8, 16, 20, 28, 32, 40, 44, 56, 100]
12	20	4.288	3	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40]
13	18	2.681	4	[1, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
14	17	2.041	5	[1, 2, 8, 16, 24, 32, 40, 44, 48, 52, 56, 60, 80, 100]
15	16	1.668	6	[1, 2, 4, 12, 16, 24, 28, 36, 40, 48, 52, 56, 60, 80, 100]
16	15	1.508	7	[1, 2, 4, 8, 16, 20, 24, 32, 36, 40, 48, 52, 56, 60, 80, 100]
17	14	1.563	8	[1, 2, 4, 8, 12, 20, 24, 28, 32, 40, 44, 48, 52, 56, 60, 80, 100]
18	13	1.525	9	[1, 2, 4, 8, 12, 16, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
19	12	2.080	10	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
21	11	1.679	11	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
23	10	1.800	12	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
26	9	1.716	13	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
29	8	1.524	14	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
33	7	1.526	15	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
38	6	1.715	16	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
46	5	1.644	17	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
57	4	1.509	18	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
76	3	1.553	19	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
114	2	1.485	20	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]
228	1	1.523	21	[1, 2, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 80, 100]

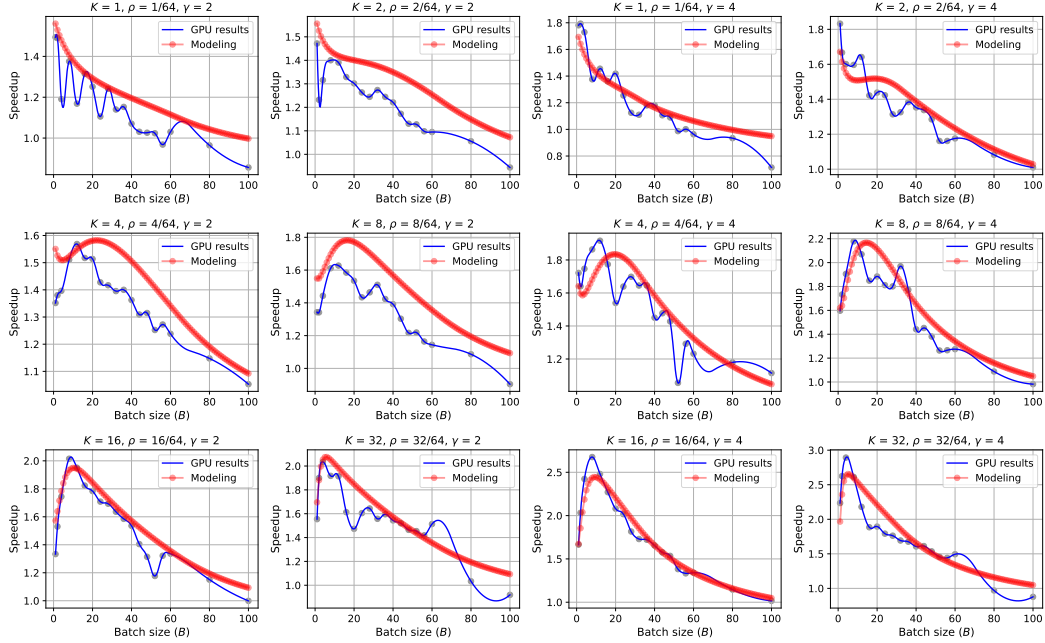


Figure 1: Comparison between GPU results and modeling with 10 measurements.

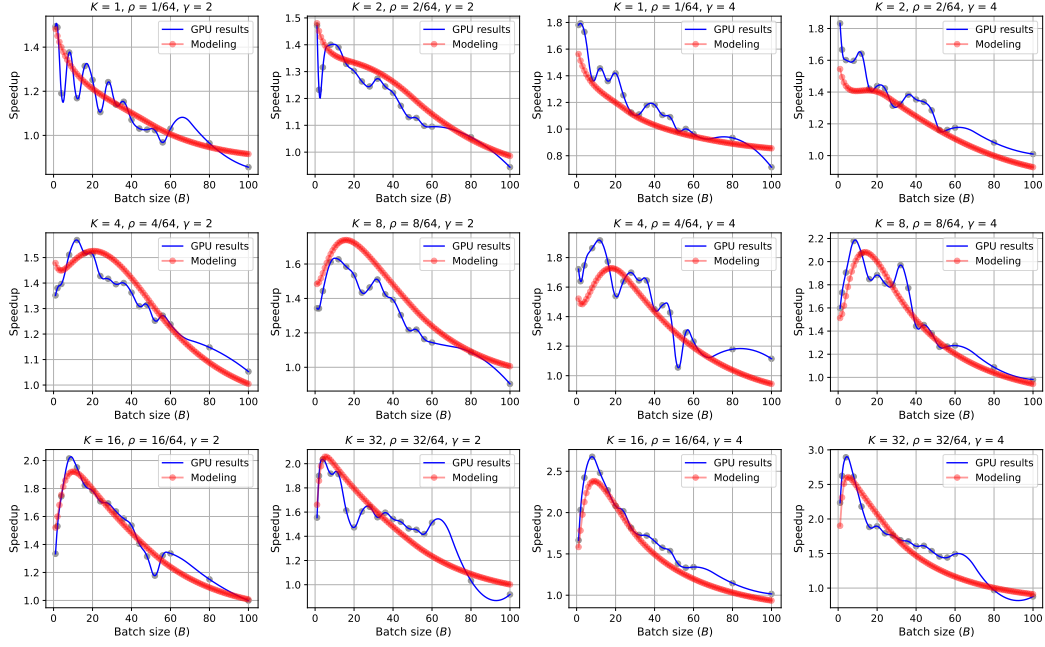


Figure 2: Comparison between GPU results and modeling with 11 measurements.

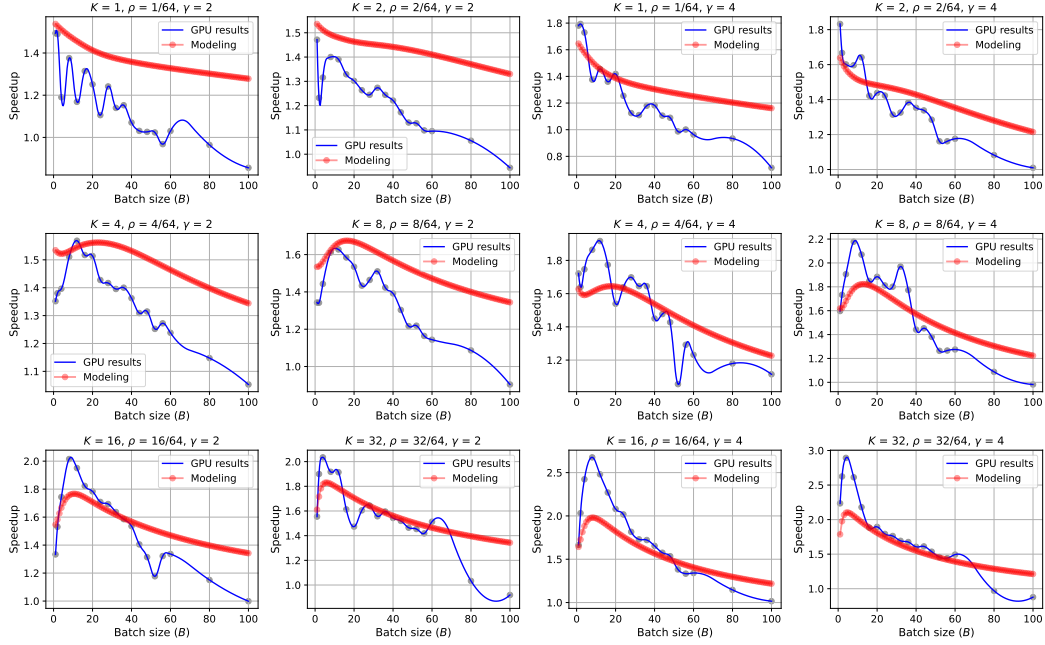


Figure 3: Comparison between GPU results and modeling with 12 measurements.

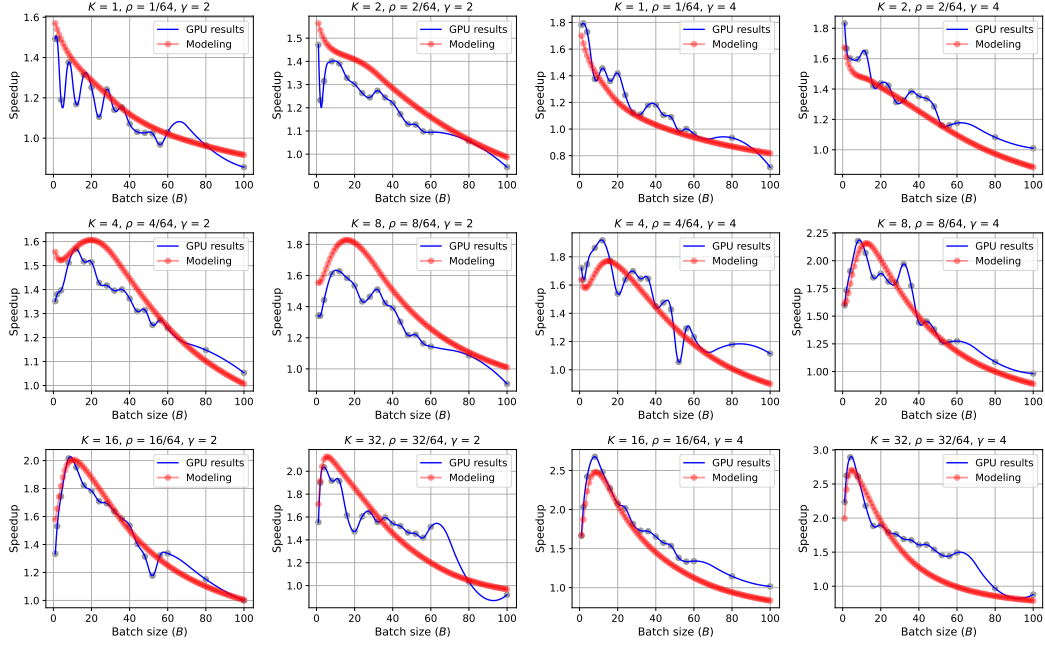


Figure 4: Comparison between GPU results and modeling with 13 measurements.

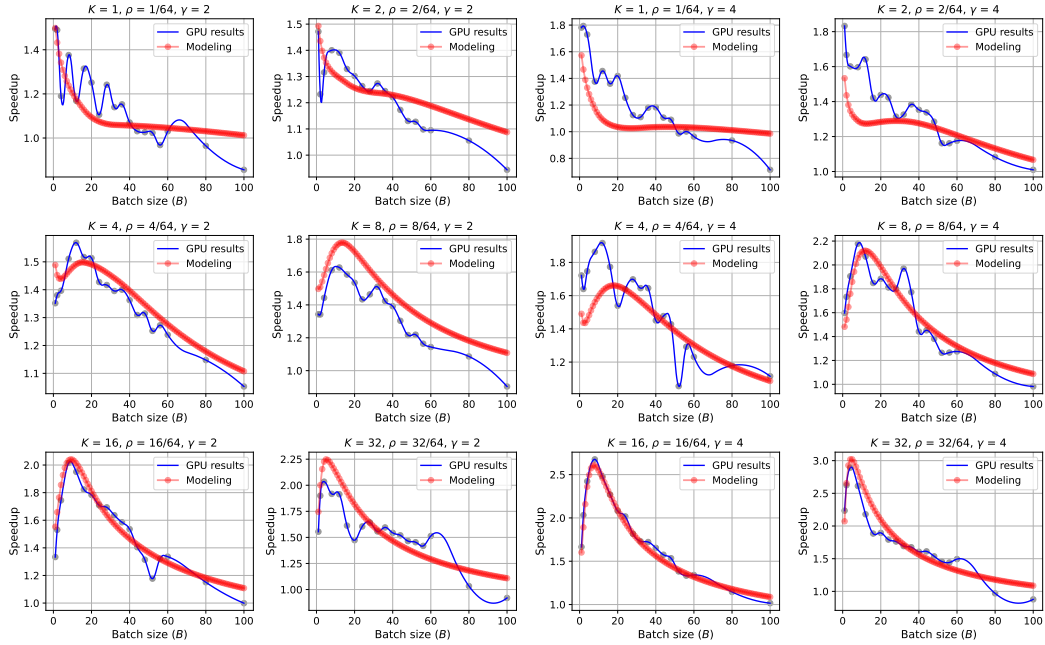


Figure 5: Comparison between GPU results and modeling with 14 measurements.

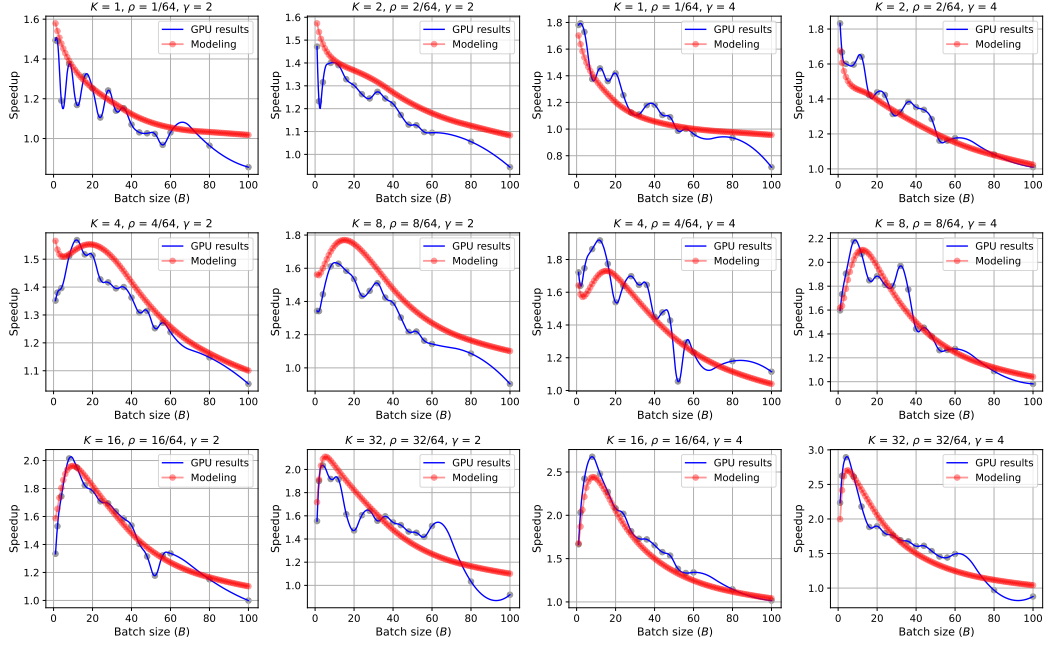


Figure 6: Comparison between GPU results and modeling with 15 measurements.

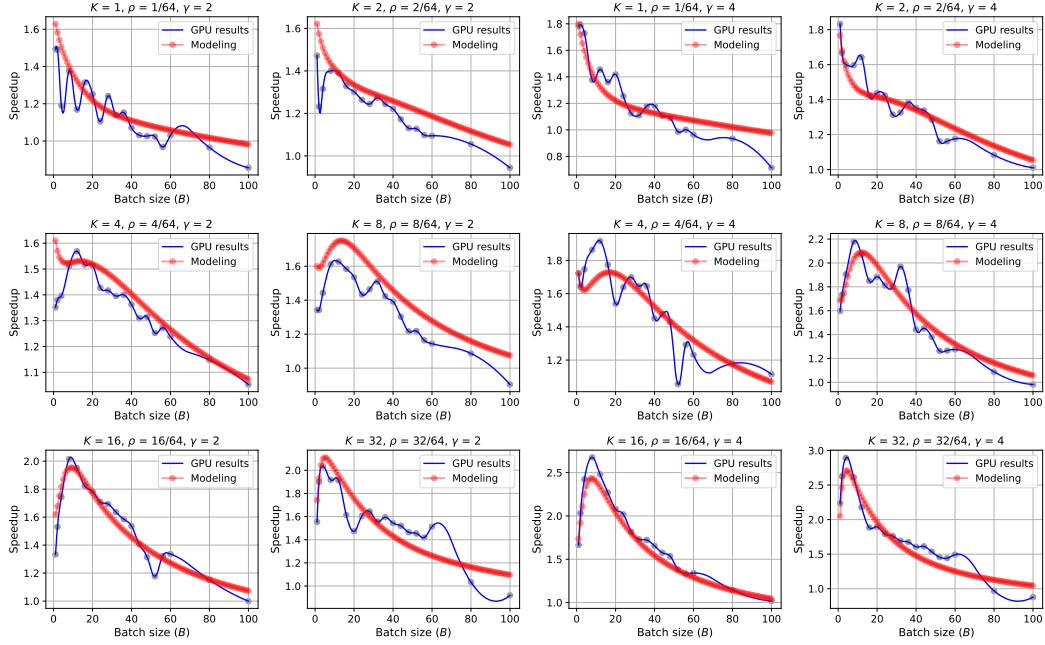


Figure 7: Comparison between GPU results and modeling with 16 measurements.

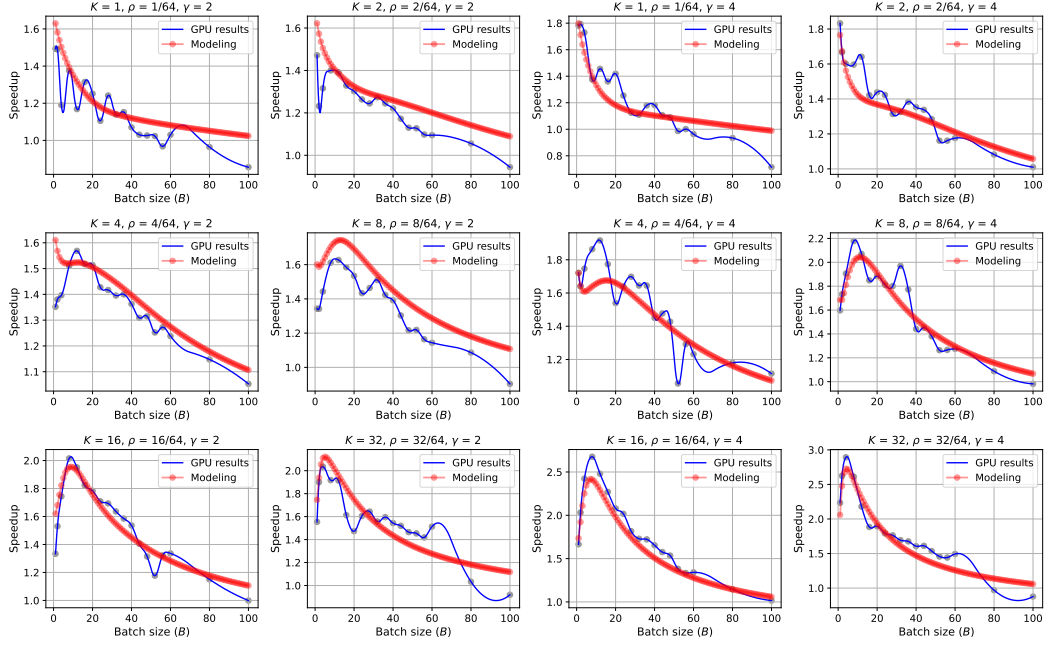


Figure 8: Comparison between GPU results and modeling with 17 measurements.

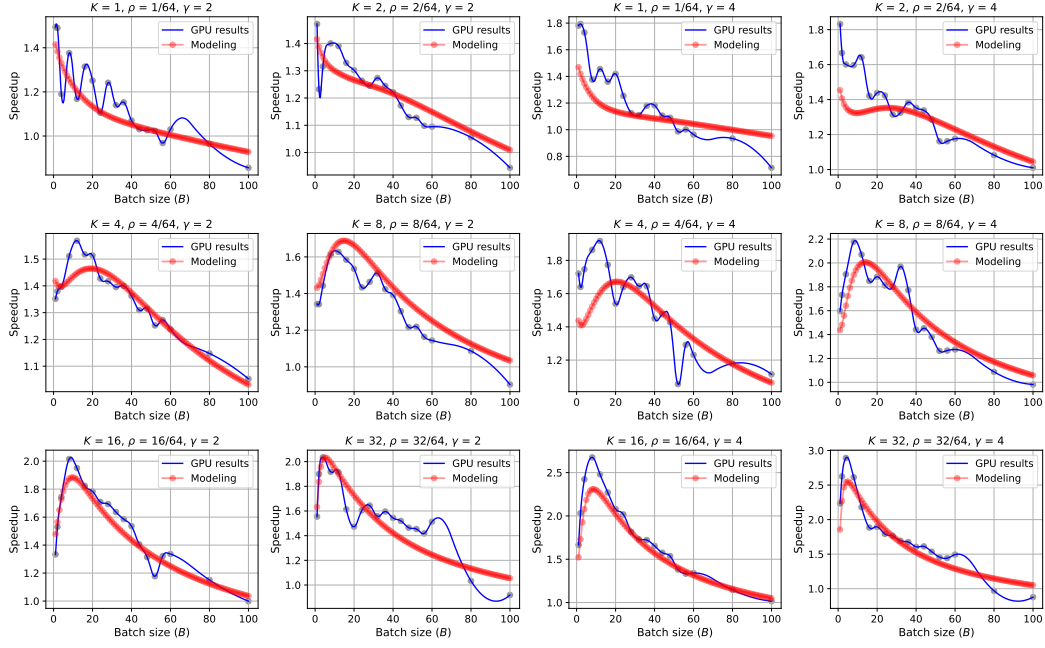


Figure 9: Comparison between GPU results and modeling with 18 measurements.

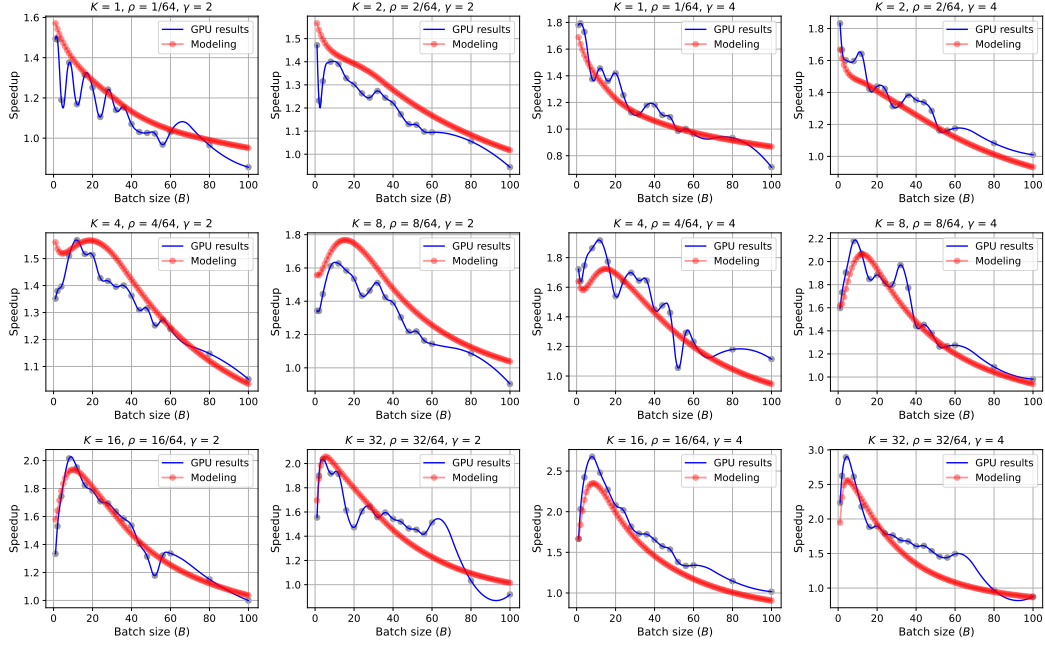


Figure 10: Comparison between GPU results and modeling with 19 measurements.

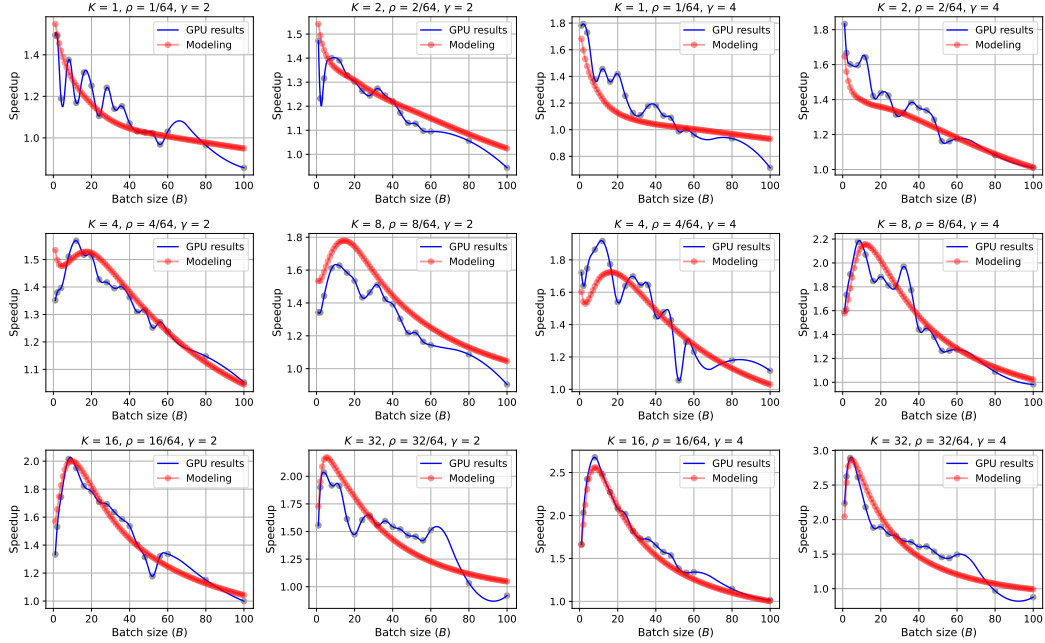


Figure 11: Comparison between GPU results and modeling with 21 measurements.



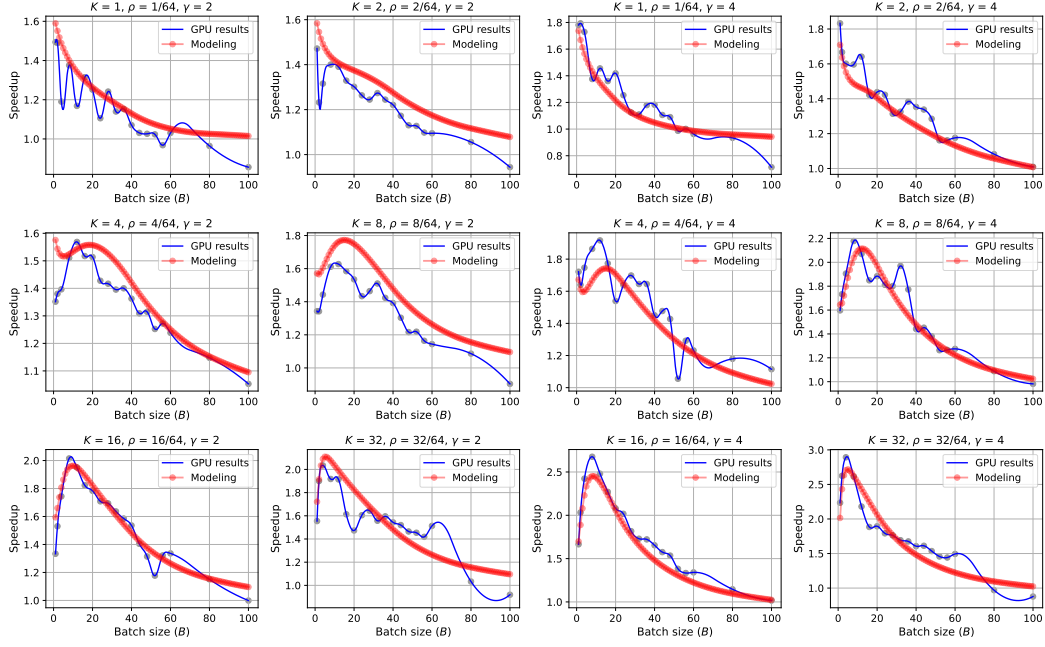


Figure 12: Comparison between GPU results and modeling with 23 measurements.

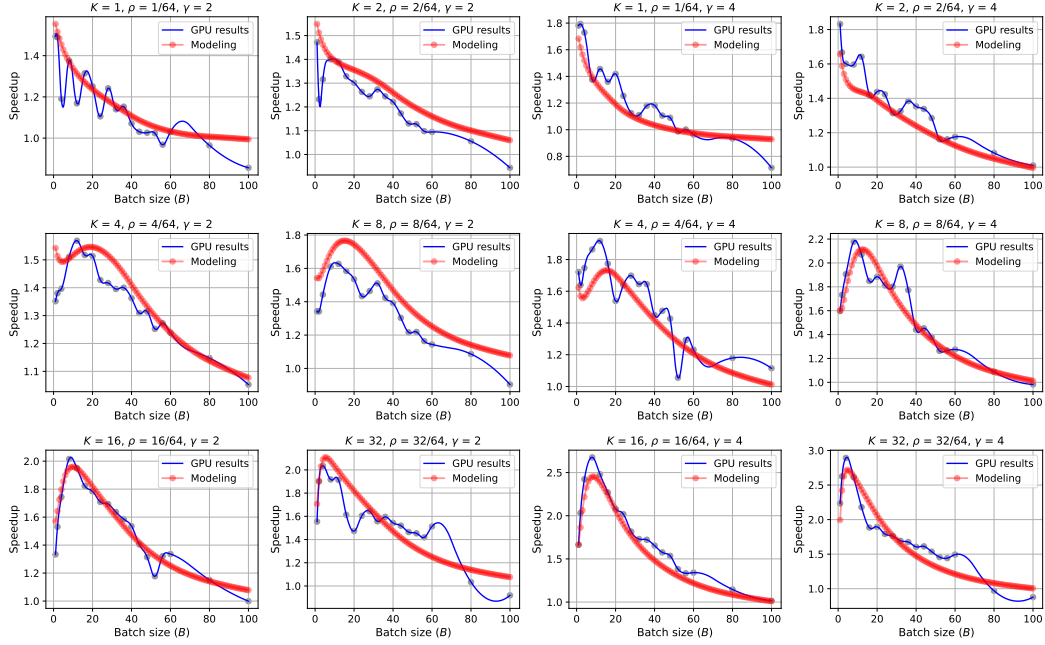


Figure 13: Comparison between GPU results and modeling with 26 measurements.

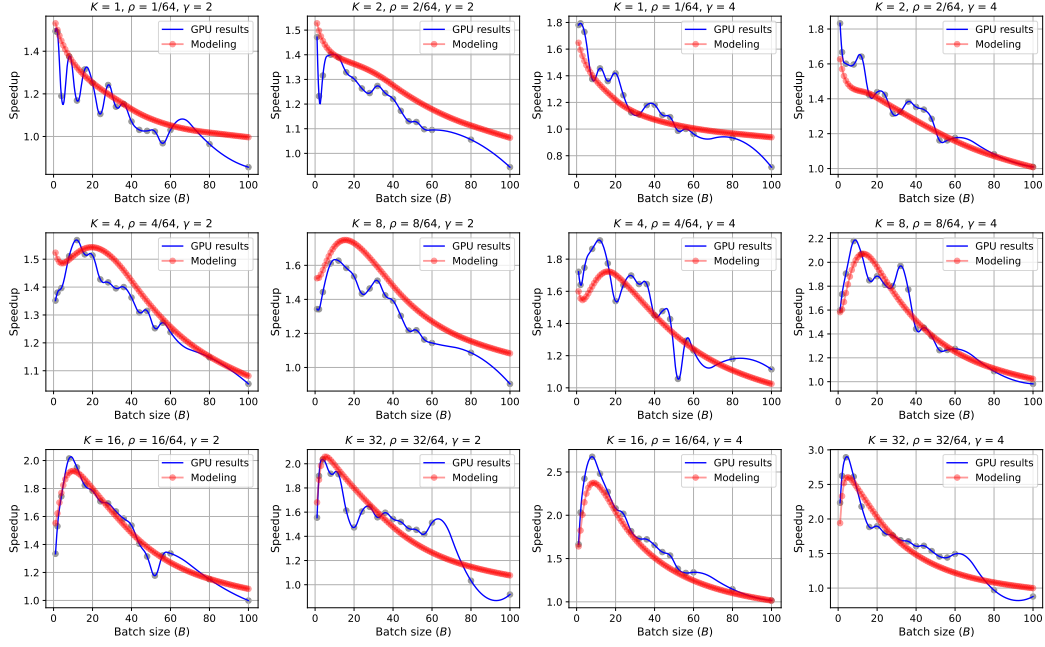


Figure 14: Comparison between GPU results and modeling with 29 measurements.

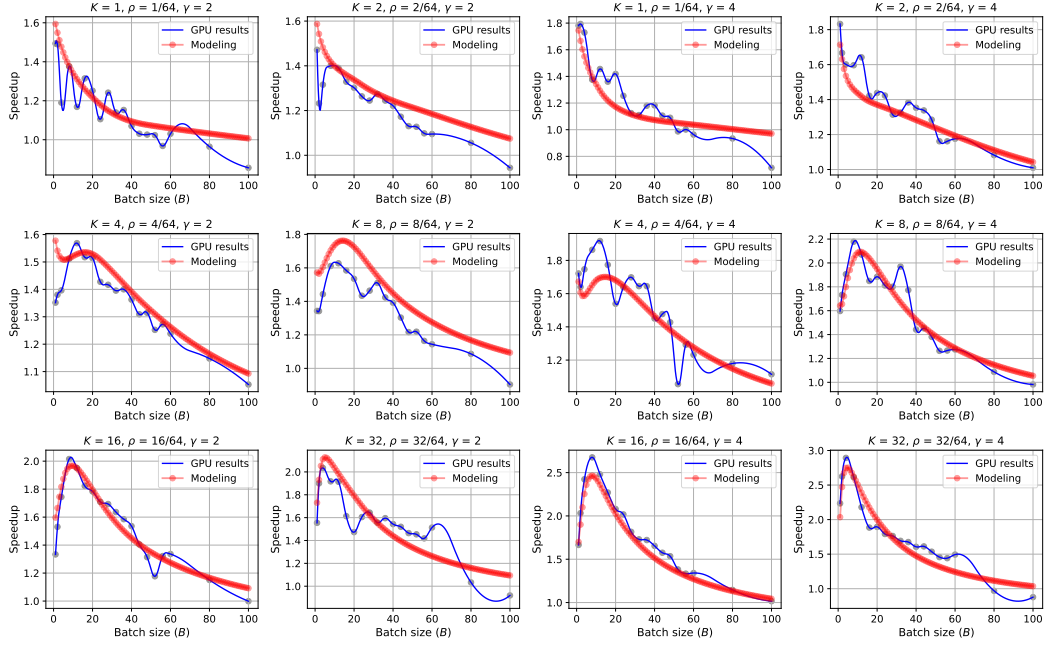


Figure 15: Comparison between GPU results and modeling with 33 measurements.

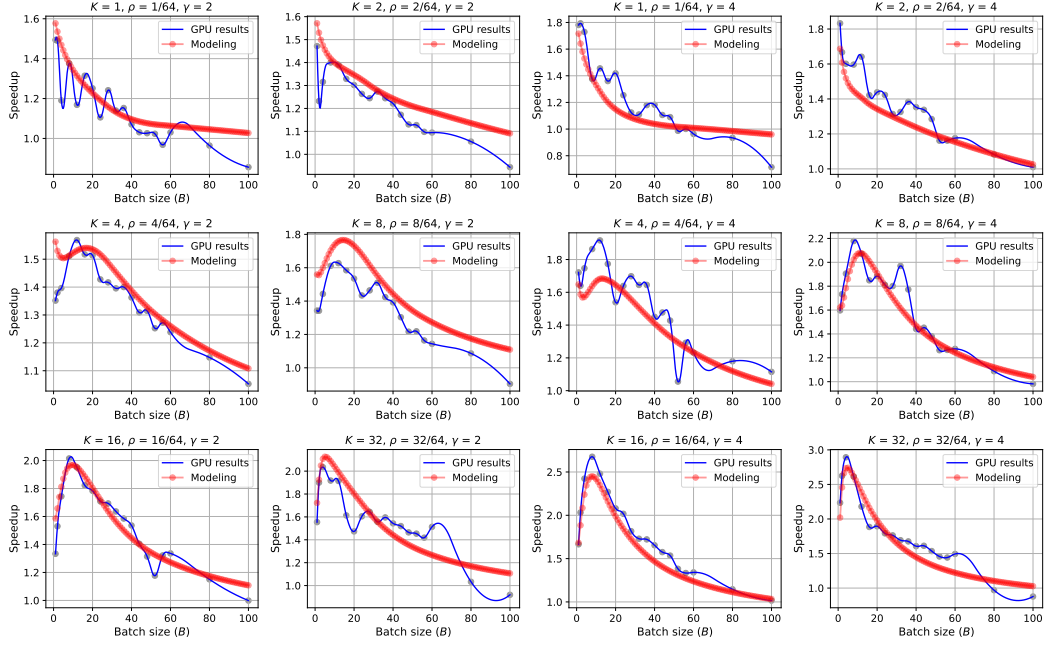


Figure 16: Comparison between GPU results and modeling with 38 measurements.

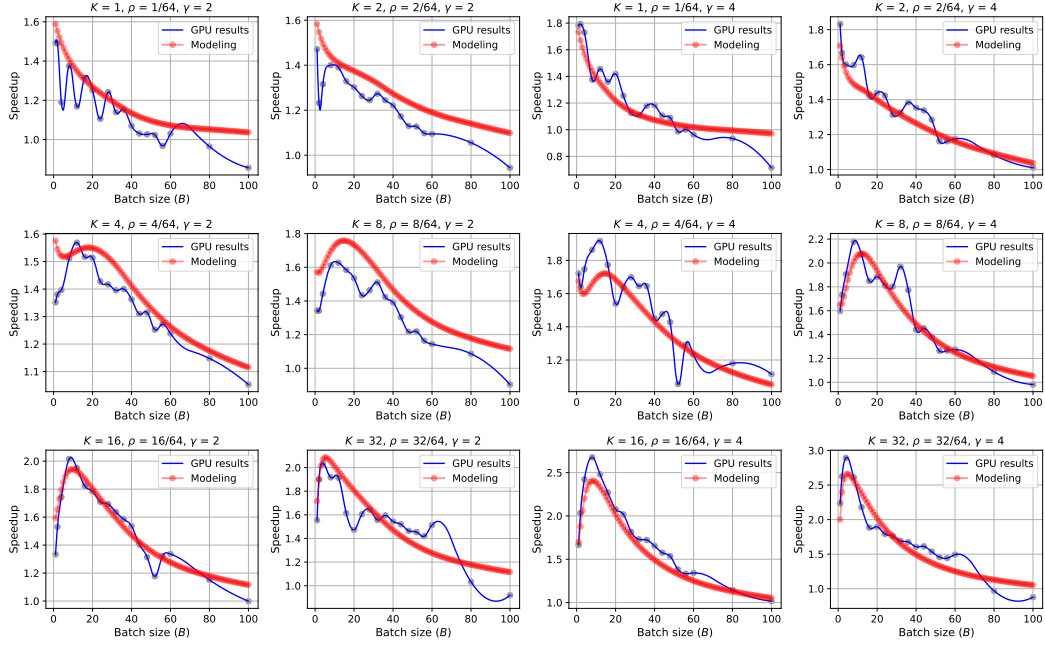


Figure 17: Comparison between GPU results and modeling with 46 measurements.

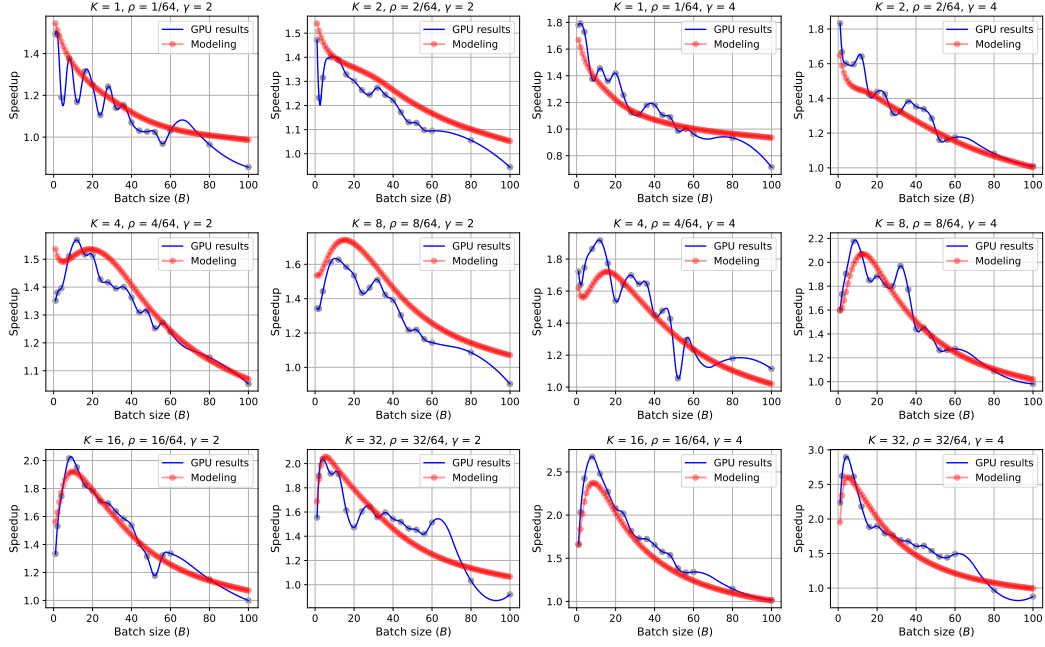


Figure 18: Comparison between GPU results and modeling with 57 measurements.

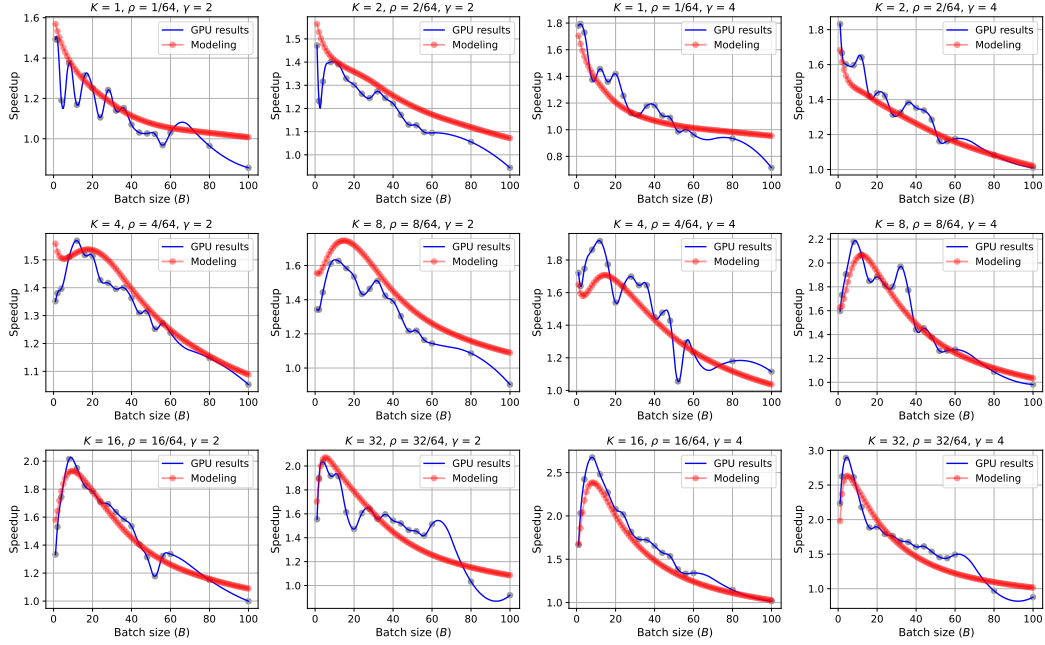


Figure 19: Comparison between GPU results and modeling with 76 measurements.

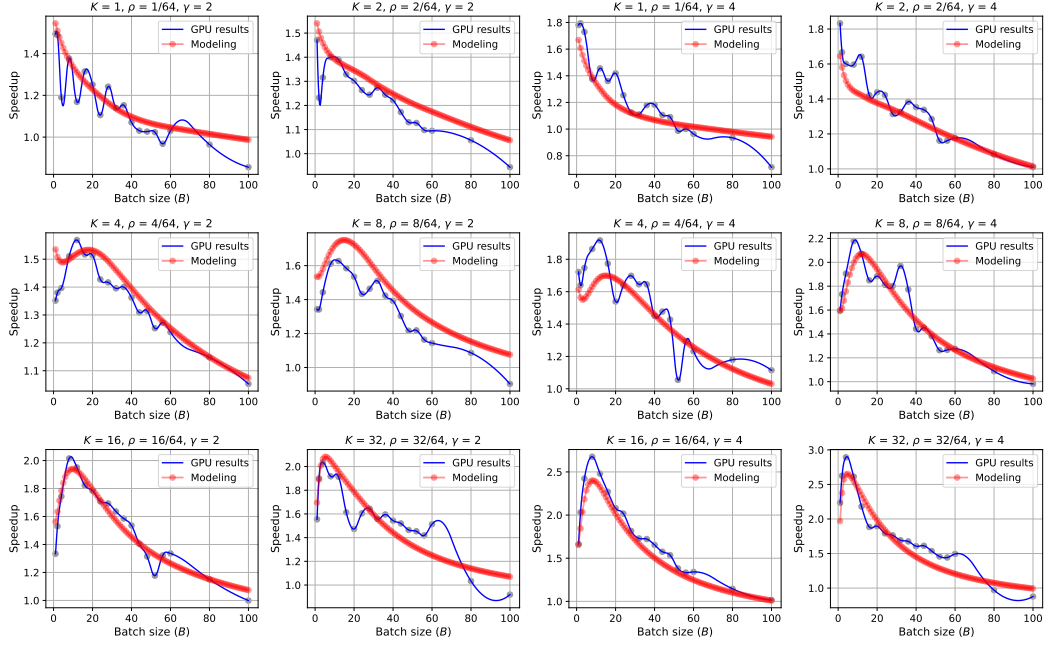


Figure 20: Comparison between GPU results and modeling with 114 measurements.

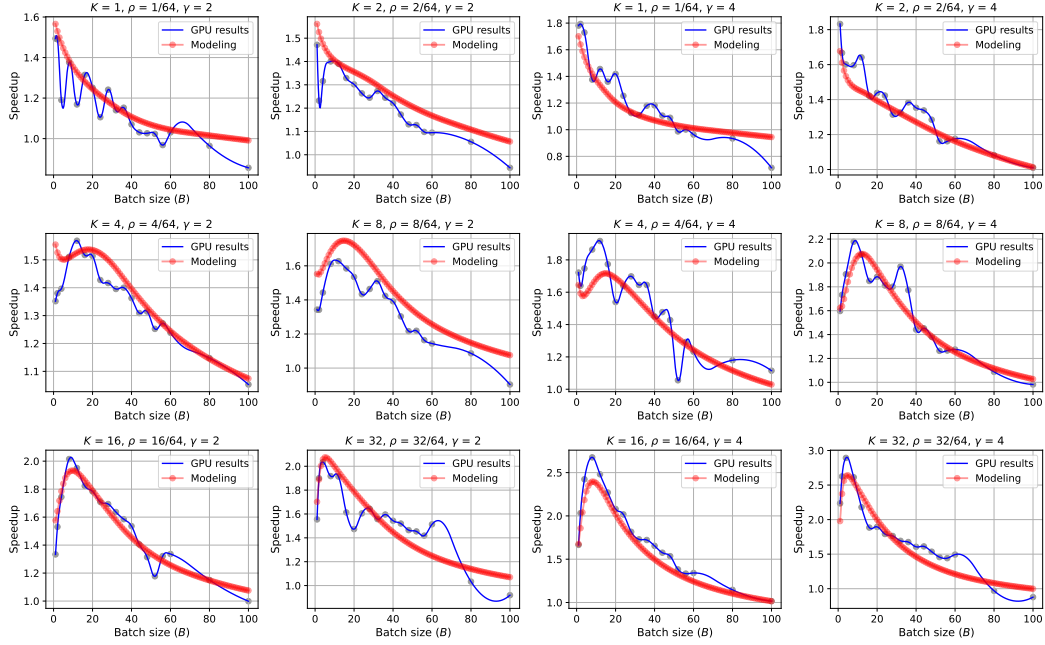


Figure 21: Comparison between GPU results and modeling with 228 measurements.